CICS® Transaction Server for VSE/ESA™

Shared Data Tables Guide

Release 1

SC33-1668-00

CICS® Transaction Server for VSE/ESA™

Shared Data Tables Guide

Release 1

SC33-1668-00

Note!

Before using this information and the product it supports, be sure to read the general information under “Notices” on page 97.

First Edition (June 1999)

This edition applies to Release 1 of CICS Transaction Server for VSE/ESA, program number 5648-054, and to all subsequent versions, releases, and modifications until otherwise indicated in new editions. Make sure you are using the correct edition for the level of the product.

The CICS for VSE/ESA Version 2.3 edition remains applicable and current for users of CICS for VSE/ESA Version 2.3.

This book is based on the *Data Tables Guide*.

Order publications through your IBM representative or the IBM branch office serving your locality.

At the back of this publication is a page entitled “Sending your comments to IBM”. If you want to make any comments, please use one of the methods described there.

Contents

Preface

What this book is about

This book gives information about CICS shared data table services.

Who should read this book

This book is for anyone who is involved with CICS shared data tables in one or more of the following areas:

- Planning
- Application programming
- Resource definition
- Customization
- Operations
- Problem determination

What you need to know to understand this book

You need to have a good understanding of the area of CICS that you are responsible for.

You should understand how the following terms are used in this book:

Browse request
A STARTBR, RESETBR, ENDBR, READNEXT, or READPREV application programming command.

Gap
When records are omitted from a CICS-maintained data table but are present in the source data set, the range of omitted keys is referred to as a "gap" in the key sequence.

Imprecise key
A record key that is specified with the GENERIC or GTEQ option in an application programming command.

Precise key
A record key that is specified without the GENERIC or GTEQ option in an application programming command.

Update request
A WRITE, DELETE, READ UPDATE, or REWRITE application programming command.

Notes on terminology

The terms listed in Table 1 are commonly used in the CICS Transaction Server for VSE/ESA Release 1 library. See the *CICS Glossary* for a comprehensive definition of terminology.

Table 1 (Page 1 of 2). Commonly used words and abbreviations

Term	**Definition (and abbreviation if appropriate)**
$(the dollar symbol)	In the character sets and programming examples given in this book, the dollar symbol ($) is used as a national currency symbol and is assumed to be assigned the EBCDIC code point X'5B'. In some countries a different currency symbol, for example the pound symbol (£), or the yen symbol (¥), is assigned the same EBCDIC code point. In these countries, the appropriate currency symbol should be used instead of the dollar symbol.
BSM	BSM is used to indicate the basic security management supplied as part of the VSE/ESA product. It is RACROUTE-compliant, and provides the following functions: • Signon security • Transaction attach security
C	The C programming language
CICSplex	A CICSplex consists of two or more regions that are linked using CICS intercommunication facilities. Typically, a CICSplex has at least one terminal-owning region (TOR), more than one application-owning region (AOR), and may have one or more regions that own the resources accessed by the AORs
CICS Data Management Facility	The new facility to which all statistics and monitoring data is written, generally referred to as “DMF”
CICS/VSE	The CICS product running under the VSE/ESA operating system, frequently referred to as simply “CICS”
COBOL	The COBOL programming language
DB2 for VSE/ESA	Database 2 for VSE/ESA which was previously known as “SQL/DS”.

Table 1 (Page 2 of 2). Commonly used words and abbreviations	
Term	**Definition (and abbreviation if appropriate)**
ESM	ESM is used to indicate a RACROUTE-compliant external security manager that supports some or all of the following functions: • Signon security • Transaction attach security • Resource security • Command security • Non-terminal security • Surrogate user security • MRO/ISC security (MRO, LU6.1 or LU6.2) • FEPI security.
FOR (file-owning region)—also known as a DOR (data-owning region)	A CICS region whose primary purpose is to manage VSAM and DAM files, and VSAM data tables, through function provided by the CICS file control program.
IBM C for VSE/ESA	The Language Environment-conforming version of the C programming language compiler. Generally referred to as “C/VSE”.
IBM COBOL for VSE/ESA	The Language Environment-conforming version of the COBOL programming language compiler. Generally referred to as “COBOL/VSE”.
IBM PL/I for VSE/ESA	The Language Environment-conforming version of the PL/I programming language compiler. Generally referred to as “PL/I VSE”.
IBM Language Environment for VSE/ESA	The common runtime interface for all LE-conforming languages. Generally referred to as “LE/VSE”.
PL/I	The PL/I programming language
VSE/POWER	Priority Output Writers Execution processors and input Readers. The VSE/ESA spooling subsystem which is exploited by the report controller.
VSE/ESA System Authorization Facility	The new VSE facility which enables the new security mechanisms in CICS, generally referred to as “SAF”
VSE/ESA Central Functions component	The new name for the VSE Advanced Function (AF) component
VSE/VTAM	“VTAM”

Road map

Table 2. Getting started road map

If you want to...	Refer to...
Obtain an overview of shared data tables	Chapter 1, “Introduction” on page 1
Learn specifically about CICS-maintained data tables	Chapter 2, “CICS-maintained data table” on page 9
Learn specifically about user-maintained data tables	Chapter 3, “User-maintained data table” on page 11
Find out how to plan for shared data tables	Chapter 4, “Planning to use data tables” on page 15
Start application programming for shared data tables and learn about cross-memory services and SDT services	Chapter 5, “Application programming for shared data tables” on page 27
Define your data tables	Chapter 6, “Managing SDT resource definitions” on page 35
Use the shared data table sevices user exits	Chapter 7, “Using the CICS-supplied global user exits” on page 41
Find out about the operational aspects of shared data tables	Chapter 8, “Using shared data tables services” on page 47
Understand shared data tables trace and dump information	Chapter 9, “Investigating problems” on page 59
Study the sample user exit programs	Appendix A, “Sample user exit programs” on page 69

Chapter 1. Introduction

The CICS® Shared Data Table (SDT) facility is an extension of the CICS file management services.

This chapter introduces shared data tables under these headings:

The concept of shared data tables

Shared data tables improve performance by:

- Using VSE/ESA® cross-memory services instead of CICS function shipping to share a file of data between two or more CICS systems that are running in different partitions under the same VSE/ESA operating system.
- Accessing data from memory instead of from DASD.
- Accessing a file of data from memory using services integrated within CICS file management instead of using VSAM services and a local shared resource (LSR) pool.

SDT completely replaces and extends the original data table services that were provided as part of the base product in CICS/VSE® Versions 2.2 and 2.3. Under SDT, all files that are defined as data tables can potentially be shared using cross-memory services. No changes are required to the file definitions for existing data tables.

The use of cross-memory services is one of the major enhancements to data table services that are included in the SDT facility. This enhancement improves the performance of applications that currently use function shipping and makes file sharing feasible for applications that cannot accept the performance overhead of function shipping.

The other major enhancement is that nearly all read requests are supported for use with data tables. This enhancement extends the use of data tables to applications that include:

- Browse requests
- Read requests that use an imprecise key

See "What you need to know to understand this book" on page v for the definition of application programming terms used in this book.

Description of data tables

A CICS file is a representation of a data set on DASD. If you specify that the file is to use data table services, CICS copies the contents of the data set into a VSE/ESA data space when the file is opened and uses that copy whenever possible.

Because of the way that the data table services access the records, they can be used only with a VSAM key-sequenced data set (KSDS). The KSDS is called the *source data set*. The copy in memory is called the *data table*. The process of copying the records is called *loading* the data table.

When the file is read by a CICS application, the record is normally retrieved from the data table. When the file is updated by a CICS application, the effect depends on the type of data table that you have defined for the file.

CICS data table services support two types of data table:

- CICS-maintained data table (CMT)
- User-maintained data table (UMT)

CICS-maintained data table

The records of a CMT are automatically reflected in the source data set; when you update the file, CICS changes both the source data set and the data table.

A CMT is easy to implement—you need to know little about the data table services, you do not need to change your existing application programs, and full recovery support of the file is retained. CMTs are discussed in more detail in Chapter 2, "CICS-maintained data table" on page 9.

User-maintained data table

The records of a UMT are not automatically reflected in the source data set; when you update the file, CICS changes only the data table.

A UMT lets you optimize the benefits of using a data table by allowing you to eliminate activity on the source data set, for update requests as well as read requests.

A small number of file operations are not supported for UMTs. Thus, you might need to make minor changes to existing application programs. Also, recovery of the file is supported after a transaction failure, but not a system failure. UMTs are discussed in more detail in Chapter 3, "User-maintained data table" on page 11.

Data table sharing environment

The environment for sharing a data table is the same as for any CICS file: one CICS region, known as a *file-owning region* (FOR), owns the data table. Any other CICS region that uses the data table is known as an *application-owning region* (AOR). In the FOR, the file is known as a *local file* and, in the AOR, the file is known as a *remote file*.

In the context of shared data tables, the FOR is also known as a *server* and the AOR is also known as a *requester*.

The same region can be both an FOR for some data tables and an AOR for others.

For information about these intercommunication concepts, see the *CICS Intercommunication Guide*.

Source data set

The source data set must be a base VSAM KSDS, not an alternate index. However, updates made to the KSDS using an alternate index are reflected in a CMT.

The VSAM definition of the KSDS supplies the values for maximum record length and key length.

Data space

The data table records are stored in a VSE/ESA data space, whether the data table is to be shared by more than one CICS region or not. A separate data space is used for each VSE partition; it is obtained when the first file that is defined as a data table is opened in the partition; it is used by all CICS data tables that are owned by that partition; and it is retained until the shutdown of CICS in the partition.

The data space storage that is used by the data table is freed when the file is closed in the FOR. This storage is made available for reuse in such a way that the integrity of any AOR that was using the data table is protected.

Global user exits

Three global user exits are provided to extend the normal processing done by data table services:

- XDTRD, to select the records that are copied to the data table during loading when the file is opened. For a UMT, it can also be used to modify the records.
- XDTAD, to select the records that are copied to the data table when new records are added to the file.
- XDTLC, to perform processing at the end of the loading operation.

These user exits are fully described in Chapter 7, “Using the CICS-supplied global user exits” on page 41.

Benefits of shared data tables

SDT offers many additional benefits over the data table services that were included as part of CICS/VSE 2.2 and 2.3. For example:

- Very large reductions in path length can be achieved for remote accesses because function shipping is avoided for most read and browse requests.
- When cross-memory services are used, the requests are processed by the AOR, thus freeing the FOR to process other requests. This increases multiprocessor exploitation.

- Increased security of data is provided because the record information in shared data tables is stored outside the CICS region and is not included in CICS system dumps (either formatted or unformatted).
- For CMTs, all forms of non-update, keyed access (including browse requests and imprecise-key read requests) are processed by reference to the data table.
- For UMTs, all forms of non-update, keyed access (including browse requests and imprecise-key read requests) are supported.
- Any number of files referring to the same source data set that are open at the same time can retrieve data from the one CMT.
- An enhancement to the XDTRD user exit allows you to skip over a range of records while loading the data table.

How a remote file is accessed

This section illustrates the differences between using function shipping and using shared data table services to access a CICS file in another CICS region.

Using function shipping

Figure 1 shows the use of function shipping to access a data set owned by another CICS region.

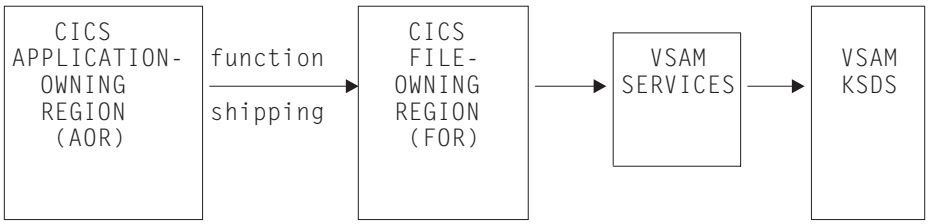


Figure 1. Data access using function shipping

Using shared data table services

Figure 2 on page 5 shows how a number of AORs can use cross-memory services to execute reads or browses, using shared data table services in an FOR to access the data table. (Function shipping is used for update requests and for any request that needs to access the source data set, in the same way as shown in Figure 1.)

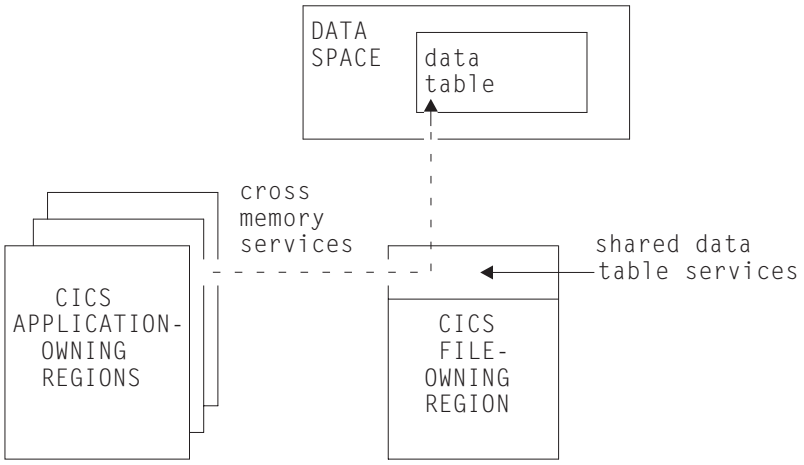


Figure 2. Data access using shared data table services. This diagram shows read-only access only.

How a data table is shared

SDT performs two operations—LOGON and CONNECT—in order to establish a data table for sharing.

LOGON

When the first file that is defined as a data table is opened in an FOR, the FOR attempts to register itself as an SDT server. This operation is performed automatically and is known as an **SDT LOGON**. The opening of the file can be caused by the FOR or by the AOR that first accesses the file.

Regardless of whether the LOGON is successful or not, the file is opened and the data table is loaded. If the LOGON is successful, all other CICS regions in the VSE/ESA operating system are notified that the data table is available.

- If the LOGON fails because of a permanent condition (such as cross-memory services not being available to this CICS region), no further LOGON attempts are made during the CICS run.
- If the LOGON fails because of a potentially transient condition, another LOGON attempt is made the next time that a file that is defined as a data table is opened. This type of condition includes:
  - Failing a security check
  - Failing to obtain storage
  - Failing to load a program
- If a CICS region's LOGON requests are rejected because they failed a security check, security violation messages might be issued each time a file that is defined as a data table is opened.

After an FOR logs on successfully, it remains in that state for the rest of the CICS run; no more LOGON requests are issued.

CONNECT

When an AOR with SDT issues a read request (or starts a browse sequence) for a remote file, SDT attempts to establish a connection to a data table for that file. If the FOR is registered as an SDT server, SDT establishes a cross-memory link from the AOR to the FOR (subject to security checks) and calls the SDT server to ask whether there is an available data table for the file. If there is, a connection is made between the AOR and the data table. This operation is performed automatically, and is known as an **SDT CONNECT**.

If the CONNECT is successful, cross-memory services are used, whenever possible, to access the file while the connection exists.

If the CONNECT fails, the file request is function shipped exactly as it would have been in the absence of SDT. The action taken for subsequent remote file requests depends on the type of failure:

- If the CONNECT fails because of a permanent condition (such as cross-memory services not being available to this CICS region), no further CONNECT attempts are made during the CICS run.
- If the CONNECT fails because of a potentially transient condition that is not under the control of the file owner, another CONNECT attempt is made for the next suitable request after approximately ten minutes have elapsed. This type of condition includes:
  - Failing a security check
  - Failing to obtain storage
  - Failing to load a program
- If a region's CONNECT requests are rejected because they failed a security check, the related security violation messages might be issued at 10-minute intervals.
- If the CONNECT fails because of a potentially transient condition that is under the control of the file owner, another CONNECT attempt is made for the next suitable request following notification that at least one new file is available for shared access on the VSE system. This type of condition includes:
  - File owner is not logged on as a server
  - File is not associated with a data table
  - File is disabled, although associated with a data table
  - File is closed, although defined as a data table

  After an AOR connects to a remote file successfully, it remains connected unless one of the following events occurs:

  - The AOR deletes its remote file definition.

    In this event, the connection is broken immediately.

  - The FOR closes or disables the file.

    In this event, the disconnection is scheduled at the next non-update request and occurs after all current browse sequences have terminated, as described in "Disconnection" on page 30.

  If these events are later reversed, a valid connection is established in the same way as before.

Notification that a new file is available for shared access

When a data table is opened by an FOR, it becomes available for CONNECT attempts at the start of loading for a CMT, or at the completion of loading of a UMT. Other CICS regions are notified that a data table has become available. Notification is also made when a data table (or a file that uses a CMT) is enabled, having been previously disabled.

Security

To provide security for a data table when cross-memory services are used, SDT must ensure that:

- The FOR cannot be impersonated. This is prevented by checking at LOGON time that the FOR is allowed to log on with the specified generic *applid* of the CICS system.
- An AOR cannot gain access to data that it is not supposed to see. This is prevented by checking at CONNECT time that the AOR is allowed access to the FOR and, if file security is in force, that the AOR is allowed access to the requested file.

These security checks are performed by using the system authorization facility (SAF) to invoke the security manager.

Note: A CICS region is still able to use data tables locally even if it does not have authority to act as a shared data table server.

SDT reproduces the main characteristics of function-shipping security that operate at the region level, but the following differences should be noted:

- SDT does not provide any mechanism for the FOR to perform security checks at the transaction level (the equivalent of ATTACHSEC(IDENTIFY) or ATTACHSEC(VERIFY)). Therefore, if you consider that the transaction-level checks performed by the AOR are inadequate for some files, you must ensure that those files are not associated with data tables in the FOR.
- SDT does not support preset security.
- SDT does not pass any installation parameter list (INSTLN) information to the security user exits.

For a description of the steps required to implement SDT security, see the *CICS Security Guide*.

Chapter 2. CICS-maintained data table

If a file is defined as a CMT, the source data set and the data table are treated by CICS as a single entity. This means that:

- Changes to the file are made to both the source data set and the data table.
- If another file is defined to use the same source data set, changes that are made by that file to the source data set are also made to the data table.
- If another file is defined to use the same source data set, records can be retrieved by that file from the data table.

This chapter discusses CMTs under:

CMT application programming

All CICS file control commands can be used in applications that access a CMT. This means that the benefits of data tables can be obtained immediately without any changes to existing applications.

CICS uses the data table to perform most read requests. Other requests might need to access the source data set. See Chapter 5, "Application programming for shared data tables" on page 27 for more information.

CMT resource definition

You define a file as a CMT by one of the following methods:

- CEDA DEFINE FILE command
- DFHFCT macro

Also, you can change the definition of an existing file by:

- EXEC CICS SET FILE command
- CEMT SET FILE command

Only the base VSAM cluster can have a CMT based on it. Read requests using alternate index paths do not use the data table, but changes to the source data set using alternate index paths are reflected in the data table.

After a file that is defined as a CMT has been opened, any other non-UMT file (whether defined as a CMT or not) that names the same source data set in its definition automatically uses the same data table. If any of these other files are defined as CMTs, message DFHFC0934 is issued to the console when they are opened. This is not an error situation; the files are opened and use the existing data table whenever possible.

Either fixed-or variable-length record format can be specified for a CMT. The maximum record length that is supported by SDT is 32KB. This length exceeds that supported by CICS file management, which thus imposes the actual limit. See

the topic dealing with lengths of areas passed to CICS commands in the *CICS Application Programming Guide*. The maximum number of records that is supported is 16 777 215.

For more information, see Chapter 6, “Managing SDT resource definitions” on page 35.

VSAM SHAREOPTION

If you use VSAM sharing to share a source data set, there is a risk that it could be updated by a partition other than the FOR. If this happened, the data table would no longer match the source data set. To minimize this risk, the VSAM SHAREOPTION should be set to 1 or 2.

1 either one requester can have update access to the data set or many requesters can have read-only access.

2 one requester can have update access to the data set and, at the same time, many requesters can have read-only access.

A warning message is issued if the cross-region SHAREOPTION is 3 or 4, or if it is 2 but the CMT has read-only access (which means another partition might be able to update the data set).

Data integrity

A file that uses a CMT can be defined as a recoverable resource. The source data set is recovered in the normal way after a system or transaction failure:

- After a system failure, the data table is reloaded from the recovered source data set when the file is reopened.
- After a transaction failure, changes that are made to the source data set by dynamic transaction backout are also made to the data table.

Automatic journaling is supported (in the same way as for any other file) for file operations that access the source data set. File operations that do not access the source data set are not journaled.

CMT operations

CICS loads a data table by copying each record from the source data set when the file is opened. A global user exit, XDTRD, can be invoked for each record before it is copied. The user-written exit program can reject records that are not to be copied. If you are using this user exit, you should ensure that the user exit is activated before the file is opened.

For information about writing user exits, see Chapter 7, “Using the CICS-supplied global user exits” on page 41. For information about activating user exits, see “Activating user exits” on page 41.

Chapter 3. User-maintained data table

If a file is defined as a UMT, the source data set and the data table are treated by CICS as separate entities. After a UMT has been loaded, it is independent of its source data set; the source data set is not updated when the data table is updated. Thus, a UMT is particularly suited to applications that make frequent updates to data that is of a transitory nature.

If the data table and source data set are updated separately, by defining them as different files, changes to one are not automatically reflected in the other.

This chapter discusses UMTs under:

UMT application programming

If a request cannot be satisfied from a UMT, CICS does not access the source data set (as it would for a CMT); instead it returns an exceptional-condition response.

Records that were in the source data set when the data table was opened might be absent from the data table because they were not copied during loading. This could be due to suppression by the user exit XDTRD or some abnormal event such as the data table becoming full.

Some application programming requests are not supported for a UMT. They include, for example, read requests that use the UPDATE option with an imprecise key. You might need to change existing applications to avoid these requests or to handle the exceptional conditions returned by CICS. For more information, see "Application programming for a user-maintained data table" on page 28.

You can use the user exits in data table services to put only the records that you need to access in the data table; there is no possibility of the source data set being accessed for those that you do not load.

You can also use the user exit XDTRD to modify each record (by selecting only a subset of its fields, for example) when it is loaded.

UMT resource definition

You define a file as a UMT by one of the following methods:

- CEDA DEFINE FILE command
- DFHFCT macro

You can also change the definition of an existing file by the:

- EXEC CICS SET FILE command
- CEMT SET FILE command

You can load multiple UMTs from the same source data set by using a separate command or macro to define each data table and making all the definitions refer to that data set.

Although a data table must be loaded from a VSAM KSDS, an application can then copy records to a UMT from any data source that is accessible from the CICS address space. This could be a remote CICS file owned by another region. The KSDS that is used as the source data set for the data table can be empty; it is needed only to define the maximum record length and the key length and position.

Variable-length record format must be specified for a UMT

The maximum record length that is supported by SDT is 32KB. This length exceeds that supported by CICS file management, which imposes the actual limit. See the topic dealing with lengths of areas passed to CICS commands in the *CICS Application Programming Guide*. The maximum number of records supported is 16 777 215.

For more information, see Chapter 6, “Managing SDT resource definitions” on page 35.

Data integrity

A UMT can be defined as a recoverable resource. However, changes to the data table are not recorded in the system log. Thus the data table can be recovered after a transaction failure (by dynamic backout) but not after a system failure.

After a system failure, the data table is reloaded from the source data set when the file is reopened. Remember that, at the time of failure, the contents of the source data set and data table would not have been the same unless you had ensured that either:

- No change is made to either, or
- Any change is made to both.

Automatic journaling is supported only for requests that access the source data set during loading. The records that are accessed by the loading process are journaled before user exit XDTRD, and the records that are accessed due to application requests are journaled after user exit XDTRD.

UMT operations

Like a CMT, a UMT is loaded when the file is opened. However, unlike a CMT, the global user exit XDTRD can be used to both select *and modify* the records from the source data set that are included in the data table.

The user exit XDTAD can be used to select the records that are added to the table after initial loading. This user exit cannot modify the records because, as the records are written by the application, it is assumed that they are already in the format used in the data table.

If you are using these user exits, you should ensure that the user exits are activated before the file is opened.

For information about writing user exits, see Chapter 7, “Using the CICS-supplied global user exits” on page 41. For information about activating user exits, see “Activating user exits” on page 41.

Chapter 4. Planning to use data tables

The sole reason for using data tables is to take advantage of the performance benefits that they offer. This chapter discusses:

Performance benefits of using data tables

This section contains Diagnosis, Modification, or Tuning Information.

Performance of a CICS-maintained data table

If all the data and index records of a file are completely contained in a local shared resource (LSR) pool, defining the file as a CMT does not reduce DASD I/O activity. There is, however, considerable potential for reduction in processor consumption. Also, you might be able to reduce the number of buffers in the LSR pool.

If the file is not completely contained in an LSR pool, using a CMT could result in reductions in both DASD I/O activity and processor consumption.

The saving of processor consumption for a CMT, compared with a VSAM KSDS resident in an LSR pool, depends on the application usage.

Performance of a user-maintained data table

After the loading of a UMT, DASD I/O activity is eliminated from all data table operations, so the saving of CPU consumption compared with a VSAM KSDS resident in an LSR pool is considerable.

Storage use

Shared data tables provide efficient use of data in memory. This means that considerable performance benefits are achieved at the cost of some additional use of storage.

This overview of the use of storage assumes that you understand the distinction between various types of storage, such as real and virtual storage, and address space and data space storage.

SDT uses virtual storage as follows:

- Record data is stored in a data space, which is virtual storage separate from address space virtual storage. The total record data storage at loading time is basically the total size of all records (without keys, which are stored in table-entry storage) plus a small amount of control information. Data space storage is acquired in units of 2MB, and then allocated to individual tables in increments of 128KB.

If many records are increased in length after loading, or new records are added randomly throughout a large part of the file, the amount of storage will be increased, possibly up to twice the original size.

- Table-entry storage is allocated from VSE storage above the 16MB line in the address space of the CICS FOR. It is allocated in increments of 32KB.

  There is one entry for each record in the table, plus one entry for each gap in the key sequence. The size of each entry is the keylength + 9 bytes, rounded up to the next multiple of 8 bytes.

- Index storage is also allocated from VSE storage above the 16MB line in the address space of the CICS FOR. It is allocated in increments of 32KB.

  The size of this area depends on the distribution and format of the key values as well as the actual number of records, as indicated in Table 3.

Table 3. Key distribution and format

Key distribution	Key format	Bytes per record
Dense (all keys are consecutive)	binary decimal alphabetic	5.1 8.5 19
Sparse (no keys are consecutive)	decimal alphabetic	44 51
Worst possible case	-	76

- System GETVIS storage is used for some small control blocks that need to be accessed by all regions that share data tables.

Converting a file into a shared data table could lead to an increased use of real storage, but the use of real storage for VSAM LSR buffers might be reduced if few updates are made. Also, an application that currently achieves high performance by replicating read-only tables in each CICS region might be able to make large storage savings by sharing a single copy of each table.

Selecting files for use as data tables

It is not possible to lay down any exact rules about whether a file will benefit from conversion to a shared data table. Many factors need to be taken into consideration, and an analysis of the potential uses of shared data tables support should ideally be undertaken by someone with a knowledge of how the files are used by the various applications, and of the configuration of the CICS regions.

Additional sources of information that could help you to select the files include:

- File statistics. “Using statistics to select data tables” on page 18 describes how you can use statistics information as one of the inputs to the selection task.
- The LSR pool statistics.
- Trace entries.
- Monitoring data.

However, the most beneficial input to the selection process is a thorough understanding of the applications and the way in which they use the files.

If your installation is using data tables for the first time, here are some general principles to help you select files for defining as data tables.

- You should consider using CMTs first, as these are easier to implement. If you use a CMT, no changes are required to the applications. If you use a UMT, some changes might be required.
- Use a CMT if you need to ensure the integrity of the data table across a CICS restart.
- Use a CMT if you require journaling of updates. If you require journaling of all access requests, the file is not suitable as a data table.
- The EXEC interface user exits XEIIN and XEIOUT, and the file control user exits XFCREQ and XFCREQC are not invoked in the file-owning region if a request to access a data table is satisfied by cross-memory services. When selecting a file, you should ensure that successful operation of your application does not depend on any activity performed at these user exits.
- You should be aware of the security implications of sharing a data table, as described in “Security checking” on page 23.
- If a file is frequently accessed from another CICS region, or if it is accessed by many other CICS regions, or if the accesses are predominantly read requests, then the benefits of making it a data table can be very large. Remember that the performance gain for a remote file is greater than for a local file (see “Performance benefits of using data tables” on page 15).
- For a CMT, select files that have a reasonably high proportion of requests that will access only the data table (see Chapter 5, “Application programming for shared data tables” on page 27). From among those, select the files with the highest usage of these requests, in order to maximize the performance gains.

  Information on file usage can be found in the CICS statistics for file control, which are described in the *CICS Performance Guide*. Not all read requests can take advantage of the data table, so you should check the data table information in the CICS statistics report afterward to verify that the data table is being used effectively. See “Interpreting data table statistics” on page 48 for more information.
- For a UMT, select files that have a large proportion of update activity but do not require the updates to be recovered across a CICS restart (see “Data integrity” on page 12).
- Select one or two files with the best estimates. Give preference to a small file over a large file when the estimated savings are similar, because a small file will probably use less real storage.
- Monitor your real storage consumption. If your system is already real-storage constrained, using a large data table could increase your page-in rates. This in turn could adversely affect CICS system performance. Use your normal performance tools to look at real-storage usage and paging rates.
- Consider reducing the number of buffers in the LSR pool because the use of data tables could reduce the number of times that the LSR pool is used.
- You can use the user exit XDTRD to select the records included in the data table. In addition, for a UMT, you can use user exit XDTRD to modify the records. Thus you can optimize the use of virtual and real storage by storing in the data table only the data that you need.

- A very large data table might require more virtual storage than your usual limit. In this case, you can either increase the virtual storage, or use the user exit XDTRD to suppress some records. To increase the virtual storage limit, increase the VSIZE value specified at IPL time. You might also have to increase the storage allocated to data spaces by specifying a larger value on the SYSDEF DSPACE DSIZE control statement.

Using statistics to select data tables

This section covers just one of the possible inputs to the selection task—the information available from the file statistics.

Figure 3 on page 19, Figure 4 on page 20, and Figure 5 on page 21 show some extracts from a hypothetical set of file statistics, which are used in the following discussion to demonstrate how CICS statistics can aid the selection process. The statistics are displayed as they would be reported by the CICS Transaction Server for VSE/ESA offline formatting utility program, DFHSTUP. Requested file statistics are shown, but Interval or End of Day statistics would be equally suitable. The section of File "Performance Information" statistics, which reports use of VSAM strings and buffers, is not shown here.

The numbers shown in the figures are purely for the purposes of illustration, and you should not expect the statistics at your installation to resemble them at all closely. Similarly, the configuration of CICS regions and files has been chosen to highlight certain points; it is not suggested that this is either a typical or a desirable configuration.

"Interpreting data table statistics" on page 48 discusses the statistics that are reported for files that have been defined as data tables, which you can use to assess the benefits being obtained.

```
Requested Statistics Report       Collection Date-Time 12/25/99-11:51:51  Last Reset 09:00:00  Applid CICFOR    Jobname SDTGSTF1

FILES - Resource Information

  File      Dataset Name                              Dataset    DT      Time     Time     Remote    Remote   Lsrpool
  Name           Base Dataset Name (If Applicable)     Type   Indicator Opened   Closed    Name     Sysid     ID

  APPLE     CIC01.CICOWN.APPLES                         K              07:44:12  OPEN                         1
  BANANA    CIC01.CICOWN.BANANAS                        K              09:45:08  OPEN                         1
  ORANGE    CIC01.CICOWN.CITRUS                         K              10:51:10  OPEN                         2
  PEAR      CIC01.CICOWN.PEARS                          K              07:30:14  OPEN                         3

Requested Statistics Report       Collection Date-Time 12/25/99-11:51:51  Last Reset 09:00:00  Applid CICFOR    Jobname SDTGSTF1

FILES - Requests Information

  File      Get      Get Upd  Browse   Update   Add      Delete   VSAM EXCP Requests
  Name      Requests Requests Requests Requests Requests Requests   Data    Index

  APPLE     2317265     1020        0     1019       21        1    11503      310
  BANANA     536452     1674    20344     1674      908        0     2651       70
  ORANGE    2069454    98560    17831    98327     4543     2563     8511      481
  PEAR        45871    65493     6512    65493    30109      362     3773      231
  *TOTALS*  4969042   166747    44687   166513    35581     2926

Requested Statistics Report       Collection Date-Time 12/25/99-11:51:51  Last Reset 09:00:00  Applid CICFOR    Jobname SDTGSTF1

FILES - Data Table Requests Information

  File      Close    Read       Recs ¬   Adds from Add      Adds rejected Adds rejected Rewrite  Delete   Highest    Storage
  Name      Type     Requests   in Table   Reads   Requests    - Exit     - Table Full  Requests Requests Table Size Alloc(K)

DFHST0223 I There are no data table statistics to report.
```

Figure 3. CICFOR requested file statistics

```
Requested Statistics Report       Collection Date-Time 12/25/99-11:51:38  Last Reset 09:00:00  Applid CICAOR1   Jobname SDTGSTA1

FILES - Resource Information

  File      Dataset Name                              Dataset    DT      Time     Time     Remote    Remote   Lsrpool
  Name           Base Dataset Name (If Applicable)     Type   Indicator Opened   Closed    Name     Sysid      ID

  APPLE     REMOTE                                                  CLOSED   CLOSED   APPLE     CIF1        N
  BANANA    REMOTE                                                  CLOSED   CLOSED   BANANA    CIF1        N
  ORANGE    REMOTE                                                  CLOSED   CLOSED   ORANGE    CIF1        N
  ZUCCINI   REMOTE                                                  CLOSED   CLOSED   COURGET   CIA2        N

Requested Statistics Report       Collection Date-Time 12/25/99-11:51:38  Last Reset 09:00:00  Applid CICAOR1   Jobname SDTGSTA1

FILES - Requests Information

  File      Get       Get Upd  Browse   Update   Add      Delete   VSAM EXCP Requests
  Name      Requests  Requests Requests Requests Requests Requests   Data    Index

  APPLE     1158701      532        0      531       11        1        0        0
  BANANA     305641        0    19067        0        0        0        0        0
  ORANGE      58709    32854     4265    32621     1018     1001        0        0
  ZUCCINI     78914        0    14765        0        0        0        0        0
  *TOTALS*  1601965    33386    38097    33152     1029     1002

Requested Statistics Report       Collection Date-Time 12/25/99-11:51:38  Last Reset 09:00:00  Applid CICAOR1   Jobname SDTGSTA1

FILES - Data Table Requests Information

  File      Close    Read       Recs ¬   Adds from Add      Adds rejected Adds rejected Rewrite  Delete   Highest    Storage
  Name      Type     Requests   in Table   Reads   Requests    - Exit     - Table Full  Requests Requests Table Size Alloc(K)

DFHST0223 I There are no data table statistics to report.
```

Figure 4. CICAOR1 requested file statistics

```
Requested Statistics Report       Collection Date-Time 12/25/99-11:49:31  Last Reset 09:00:00  Applid CICAOR2   Jobname SDTGSTA2
FILES - Resource Information
  File      Dataset Name                              Dataset    DT      Time     Time     Remote    Remote   Lsrpool
  Name           Base Dataset Name (If Applicable)     Type   Indicator Opened   Closed     Name     Sysid      ID

  COURGET   CIC02.CICOWN.COURGET                        K              08:22:15  OPEN                          1
  LEMON     REMOTE                                                     CLOSED   CLOSED   ORANGE    CIF1       N

Requested Statistics Report       Collection Date-Time 12/25/99-11:49:31  Last Reset 09:00:00  Applid CICAOR2   Jobname SDTGSTA2
FILES - Requests Information
  File     Get      Get Upd  Browse   Update   Add      Delete   VSAM EXCP Requests
  Name     Requests Requests Requests Requests Requests Requests   Data    Index

  COURGET     78914    27469    14765    27469   336472        0     8212      481
  LEMON     2010745    65706    13566    65706     3525     1562        0        0
  *TOTALS*  2089659    93175    28331    93175   339997     1562

Requested Statistics Report       Collection Date-Time 12/25/99-11:49:31  Last Reset 09:00:00  Applid CICAOR2   Jobname SDTGSTA2
FILES - Data Table Requests Information
  File     Close    Read       Recs ¬   Adds from Add      Adds rejected Adds rejected Rewrite  Delete   Highest    Storage
  Name     Type     Requests   in Table   Reads   Requests    - Exit     - Table Full  Requests Requests Table Size Alloc(K)

DFHST0223 I There are no data table statistics to report.
```

Figure 5. CICAOR2 requested file statistics

The examples use a hypothetical configuration of three CICS regions. Most of the files used by CICS applications are owned by the file-owning region CICFOR, and the applications mostly run in the AORs CICAOR1 and CICAOR2. This discussion assumes that each of the data sets shown in the statistics reports is a VSAM base KSDS (as indicated by the `Dataset Type` of `K`), so any of them can be defined as data tables.

The CICS statistics also show you which file names in one CICS region are defined to access which file names in another CICS region. The `Remote Sysid` is the name given on the connection between the two CICS regions. In the examples, the SYSID of CICFOR is CIF2 and that of CICAOR2 is CIA2.

A file with a high read-to-update ratio

The file APPLE is used by applications that run on the application-owning region CICAOR1. It is defined in CICAOR1 as a remote file, and the file definition points to the file APPLE owned by CICFOR. This file would benefit from being redefined in CICFOR as a CICS-maintained data table because it has a high ratio of remote reads (1158701 `Get Requests` in the time period covered by the reports) to remote updates (11 adds, 1 delete and 531 updates) as seen in Figure 4 on page 20.

See the *CICS Performance Guide* for guidance on the meanings of the “FILES - Requests Information” section of a statistics report.

A file with a high proportion of remote reads

The file BANANA is updated and read on CICFOR, but is also accessed by CICAOR1. Because all the remote accesses are reads and browses, with no updates, the applications running in CICAOR1 would probably see large benefits if BANANA was defined as a data table, and the applications on CICFOR would also benefit by reading from the local data table.

A file shared by several CICS regions

From a study of the statistics in Figure 4 on page 20 it might appear that ORANGE is not an especially suitable data table candidate, as the numbers of remote retrievals from CICAOR1 (58709 `Get Requests` and 4265 `Browse Requests`) are relatively low. However, the remote file LEMON in CICAOR2 also points to ORANGE in CICFOR, so defining ORANGE in CICFOR as a shared CMT would probably benefit the performance of the applications in both AORs.

A good UMT candidate

The file COURGET owned by CICAOR2 is accessed using the file name ZUCCINI in CICAOR1. CICAOR1 only reads or browses the file; any updating is issued by the file-owning region. Also, it is known that these updates are relevant only to the day's CICS run and do not need to be retained permanently (in fact, they are deleted at shutdown). The file is therefore an excellent candidate for defining as a UMT. Then, all the updates can be made to the data table without any VSAM I/O activity, and all the remote retrievals can be made without function shipping.

Note that COURGET could not be defined as a UMT if CICAOR2 were not a CICS system with SDT support (that is, a CICS Transaction Server for VSE/ESA Release 1 system) because browse requests are issued to that file and browsing of a UMT is supported only by SDT. Also, the statistics do not show whether any of the Get requests specify the GENERIC or GTEQ options; these also are supported for a UMT only by SDT.

A rather poor candidate

The file PEAR would probably not benefit much from shared data tables support because it is not accessed remotely and has many update and browse requests. Local browsing does not offer as much benefit as either local reading or any form of remote retrieval, because VSAM browsing (apart from processing of the STARTBR command) is very efficient. This analysis, of course, does not consider the relative importance of the various file accesses: the reading might be done by critical applications, but the time taken for updates might not be important.

Other possible candidates

The preceding examples illustrate only a small sample of the possible configurations and uses of files that could benefit from shared data tables support.

You could also use shared data tables support to avoid the need to duplicate files or data tables in each CICS region. And, in addition to looking at existing files, you could consider moving files from an AOR to an FOR where this was not practical before because of the cost of file accesses using function shipping.

Security checking

The security checking that is performed by the SDT LOGON and SDT CONNECT operations is introduced in “How a data table is shared” on page 5. You should consider the implications of the security checks before sharing a file that is associated with a data table.

For information about VSE/ESA security, function-shipping security, and implementing security checking for shared data tables, see the *CICS Security Guide*.

LOGON security check

To minimize the risk that an AOR might accept counterfeit data records from a FOR which is in fact an impostor, LOGON processing includes a security check to verify that the FOR is authorized to act as a server with the specified application name. This check is always performed, even when SEC=NO is specified at system initialization.

CONNECT security checks

The security checks performed at CONNECT time provide two levels of security:

- **Bind security** allows an FOR that runs without CICS file security to be able to restrict shared access to selected AORs. (Running without file security minimizes run-time overheads and the number of security definitions.)
- **File security** can be activated in the FOR if you need a finer granularity of security checking. Then, SDT implements those checks that apply to the AOR as a whole.

SDT provides no way of implementing those security checks that an FOR makes at the transaction level when ATTACHSEC(IDENTIFY) or ATTACHSEC(VERIFY) is used with function shipping.

SDT support on different releases of CICS

To benefit from the cross-memory support provided by SDT, you must be running with SDT support in both the requesting and serving CICS systems.

If only the requesting CICS system has SDT support, there is no effect apart from the very small overhead of occasional attempts to determine whether the server system supports sharing. All requests continue to be function shipped.

If only the serving CICS system has SDT support, all requests continue to be function-shipped by the requester. The requester does, however, obtain the benefits of local data table accesses made by the server.

Preparing to use SDT support

To use SDT support, you must perform the following tasks. Some of them will already have been done for an installation that currently uses function shipping and/or data tables.

- Ensure that DFHDTSVC, DFHDTSAN, and DFHCSEOT are in the Shared Virtual Area (SVA).
- Define security authorization so that FORs can act as SDT servers and AORs can access files owned by servers, depending on the level of security required. In a single VSE image:
  - Any number of FORs can act as SDT servers
  - A single AOR can use any number of these FORs
  - A single FOR can serve any number of AORs
  - A CICS region can act as an AOR for one data table and as an FOR for a different data table
- If, for some reason, two FORs have the same APPLID, at any given time SDT ensures that only one of these FORs is used as an SDT server. However, there is nothing to prevent one FOR acting as an SDT server and another FOR, with the same APPLID, being used for function shipped requests. You should check that your operational procedures do not allow this because there is a risk that data table requests that use SDT services will not be directed to the same CICS region as requests that use function shipping.
- Define those files in the FOR which are to be data tables as either CMTs or UMTs.
- Create additional remote file definitions in the AOR if required. No changes are needed to existing remote file definitions.
- For any AOR that is to share data tables, specify ISC=YES as a system initialization parameter and define MRO or ISC links to the relevant FORs.
- Before using shared data tables, you might have to change some of your JCL statements, modify your operational procedures, or increase the size of the data space available in the VSE system.

  The data space for SDTs is acquired in units of 2MB. You must ensure that the VSIZE value specified at IPL time allows for at least this amount of virtual storage, in addition to all the other address space and data space requirements. The DSIZE value specified in the SYSDEF DSPACE statement must allow for at least this amount of data space, in addition to all the other data space requirements. For information about specifying these values, see the *VSE/ESA System Control Statements* manual.

Load modules

Table 4 shows the load modules that you must install in your CICS system from the VSE/ESA production sublibrary PRD1.BASE so that you can use SDT.

Table 4. Load modules in SDT

Load module	How loaded	Description
DFHDTINS	CICS load above the 16MB line	Initialization
DFHDTSVC	SVA mandatory	Performs all functions that need VSE/ESA authorization
DFHDTFOR	VSE LOAD above the 16MB line	Data table FOR module
DFHDTAM	VSE LOAD into subpool 252 storage above the 16MB line	Data table access manager. It includes code that is executed in cross-memory mode from an AOR
DFHDTAOR	VSE LOAD above the 16MB line	Data table AOR module
DFHDTCV	VSE LOAD into System GETVIS	Connection validation (AOR)
DFHDTXS	VSE LOAD into System GETVIS	Connection security checking (FOR)
DFHCSEOT	SVA mandatory	CICS EOJ clean up routine
DFHDTSAN	SVA mandatory	System anchor block

Storage occupancy

The total size of the modules that occupy storage above the 16MB line is about 41KB. For modules that are in System GETVIS, about 1.5KB are required for each logged-on FOR, and about 0.5KB for each AOR.

The modules are all eligible for inclusion in the SVA, but only DFHDTFOR, DFHDTAM, DFHDTAOR, and possibly DFHDTCV are used sufficiently frequently to be worth considering. DFHDTSVC, DFHDTSAN, and DFHCSEOT are mandatory for the SVA.

Chapter 5. Application programming for shared data tables

This chapter contains General-use Programming Interface and Associated Guidance Information.

This chapter describes application programming for shared data tables:

You access a data table with the same EXEC CICS file control commands that you use with any normal CICS file. These commands can be used fully with a CICS-maintained data table and with some restrictions with a UMT. General information about using these commands is in the *CICS Application Programming Guide*. For programming information, see the *CICS Application Programming Reference*.

Application programming for a CICS-maintained data table

CICS handles a CMT and its source data set as a single entity. After the data table has been loaded, CICS automatically keeps the contents of the data table and the source data set consistent; any changes that an application makes to the file are reflected in both.

All file control commands and options can be used and the use of a data table is transparent to the application programmer. The following information is provided to allow you to get the maximum benefits from your data tables.

Some commands are performed by access only to the data table (using cross-memory services for shared files), some by access only to the source data set (using function shipping for shared files), and some by access to both.

The following commands usually access only the data table:

- READ commands without the UPDATE or RBA options
- STARTBR, RESETBR, READNEXT, and READPREV commands without the RBA option

The following commands access only the source data set:

- READ commands with the UPDATE or RBA options
- STARTBR, RESETBR, READNEXT, and READPREV commands with the RBA option
- ENDBR command for a browse sequence that has accessed the source data set

The following commands might access both the data table and the source data set:

- READ and browse commands (which would usually access only the data table) that find a gap in the sequence of records in the data table. This gap might indicate that one or more records are missing from the data table because:
  - Records have been suppressed by a user exit.
  - The maximum number of records has been reached.
  - Insufficient virtual storage is available for the data table.
  - Some abnormal event has occurred.
- READ, READNEXT, and READPREV commands for records that are currently being processed by a WRITE, REWRITE, or DELETE command. These commands need to first access the data table to determine that this situation exists.
- WRITE, REWRITE, and DELETE commands. These commands are always executed in the FOR, where they first update the source data set. If successful, a corresponding change to the data table is attempted, using local SDT services in the FOR. In the case of a WRITE command, the addition of the record to the data table might be rejected by the XDTAD user exit or might fail because the data table is full or insufficient virtual storage is available.

Using a CMT during loading

It is possible to use a CMT while it is being loaded. If the required record has already been loaded, processing of the request is handled in the normal way. If the record has not yet been loaded, the following is done:

- For a READ command, the record is read from the source data set and returned to the application program. It is added to the data table when the normal loading sequence reaches it.
- For a WRITE command, the record is added to the source data set and the data table (if not suppressed by the user exit XDTAD).
- For a REWRITE or DELETE command, the change is applied to the source data set which is then reflected in the data table by the normal loading process.

Application programming for a user-maintained data table

CICS handles a UMT and its source data set as separate entities. When loading is complete, all file control commands that access the file name are performed only on the data table.

There are some restrictions on which commands and options can be used. There are also some exceptional conditions that are unique to UMTs. These restrictions and conditions are described below.

The following commands are not supported; they return the INVREQ condition and a value of 44 in the EIBRESP2 field:

- Commands with the RBA option
- WRITE commands with the MASSINSERT option
- DELETE commands with the GENERIC option
- READ commands with the UPDATE option plus either the GENERIC or GTEQ options

The following commands are supported (using cross-memory services for remote accesses):

- READ commands with neither the RBA option nor the UPDATE option. If the record does not exist in the data table, the NOTFND condition is returned.
- STARTBR, RESETBR, READNEXT, and READPREV commands without the RBA option.
- ENDBR commands.

The following commands are supported (using function shipping for remote requests):

- WRITE commands without the RBA or MASSINSERT options. The record is added to the data table (if not suppressed by the XDTAD user exit).

  The NOSPACE condition is returned if:

  – There is not enough virtual storage to add the record to the data table.

  – The data table already contains the maximum number of records that is specified in the file definition.

  The SUPPRESSED condition is returned if the user exit XDTAD suppresses the addition of the record to the data table.
- REWRITE commands without the RBA option. The record is updated in the data table. The NOSPACE condition is returned if there is insufficient virtual storage for the updated record.
- DELETE commands without the GENERIC or RBA options. The record is deleted from the data table. The NOTFND condition is returned if the record does not exist in the data table. The NOSPACE condition is returned if the data table is recoverable and there is insufficient virtual storage for the information that CICS writes about the deleted record.

Using a user-maintained data table during loading

A UMT can be accessed only by the FOR during loading. All remote requests are function shipped to the FOR, which processes them in the same way as for a local request, as described below.

While a UMT is being loaded, you can use only non-update read requests with precise keys. If the record has already been loaded, processing of the request is handled in the normal way. If the record has not yet been loaded, the record is read from the source data set and submitted to the user exit XDTRD (if activated):

- If it is not suppressed by XDTRD, the record is added to the data table and also returned to the application program.
- If it is suppressed by XDTRD, the NOTFND condition is returned.

The LOADING condition is returned for other requests that would have been valid had loading been complete.

Using cross-memory services

Cross-memory services are used to satisfy an application programming command when all of the following conditions have been met:

- CICS must retrieve the SYSID of the target system from the file's resource definition in the AOR. This condition is met when the application programming command either specifies no explicit SYSID or specifies a SYSID which is that of the AOR itself, and the SYSID given in the file resource definition is that of the FOR.

  Within a single browse sequence, an application should not change between specifying an explicit SYSID and not specifying one, as this is likely to lead to unpredictable results.
- The serving system has logged on; that is, it has registered itself as a shared data table owner.
- The requesting system has connected to the server for the files specified on the application programming command.
- The file supports the requested function.

Note: Function shipping a request might result in "daisy chaining"; that is, the request passes through one or more intermediate CICS nodes between the region issuing the request (an AOR) and the region owning the resource (the FOR). In such cases, use of shared data tables cross-memory services is limited to the final link (from the last intermediate system to the FOR).

Connection

Commands cannot use cross-memory services until the SDT connection is made between the AOR and the remote data table. Also, if a browse sequence starts before the connection is made, all subsequent requests in the sequence use function shipping services. This is likely to occur if the connection cannot be established at the STARTBR command because the data table is not open, and the command causes the data table to be implicitly opened. The connection is then made on the next new request to the data table, but the original browse sequence continues to use function shipping services.

Disconnection

A connection remains in force until either the AOR deletes its remote file definition or the FOR closes or disables the file. The effects of close or disable are described below.

- If the FOR closes the file (with or without the FORCE option), disconnection is scheduled at the next non-update request that is issued for the file (that is, the next request to attempt to use cross-memory services to access the data table).

  The disconnection takes place as soon as all outstanding browse sequences (if any) against the file have terminated. Each browse sequence terminates either at the next browse request (and the transaction is abended with code AFCH unless the request is an ENDBR command) or when the transaction terminates.

  After the disconnection is scheduled, all requests (except any outstanding browse requests, as described above) are function shipped until a connection is re-established.

- If the FOR disables the file without the FORCE option, disconnection is scheduled at the next non-update READ or STARTBR command issued for the file, unless the FOR re-enables the file before then.

  If scheduled, disconnection takes place as soon as all outstanding browse sequences (if any) against the file have ended. Such browse sequences continue normally; they are unaffected by the disabling unless a browse of the source data set is started in the FOR in order to satisfy a request in the browse sequence (see “Disabling a data table”).
- If the FOR disables the file with the FORCE option, the effect is the same as when a file is closed except that, if the FOR re-enables the file before the AOR issues the next non-update request for the file, the disabling is not observed by the AOR and so disconnection is not scheduled.

Differences between function-shipping and cross-memory services

You should be aware of the following differences between the way requests are handled, depending on whether function-shipping or cross-memory services are used to access the data table.

Closing a data table

When function shipping is used for a browse sequence of a remote file, the file cannot be closed (except by using the FORCE option) until after the browse sequence ends.

When cross-memory services are used, it is possible for the file to be closed during the browse sequence. In this case, the transaction is ended with abend code AFCH at the next request for that file. If your applications or operational procedures rely on the quiescing of browse activity either when closing a file or at the normal shutdown of an FOR, you should review them before using a shared data table for the file.

Disabling a data table

When function shipping is used for a browse sequence of a remote file, the browse sequence, once started, can continue normally even if the file is then disabled (unless the FORCE option is used).

When cross-memory services are used, the effect is the same unless, during the browse sequence, it is necessary to function ship a STARTBR command to the FOR. This can happen if, for example, a gap in a CMT makes it necessary to browse the VSAM source data set to retrieve records. The function-shipped STARTBR command fails if the file is then disabled by a request that was issued by the FOR after the browse sequence started in the AOR. In this case, the browse sequence is unable to continue normally, so the transaction in the AOR is abended with code AFCH.

If the FORCE option is used with the disable request, all function-shipped browse requests are always terminated. If the file is re-enabled, it is possible for browse requests that use cross-memory services to continue unaffected. See the information about FORCE in “Disconnection” on page 30.

User exits

For function-shipped requests, the EXEC interface user exits XEIIN and XEIOUT, and the file control user exits XFCREQ and XFCREQC are invoked in both the AOR and FOR.

For cross-memory requests, these user exits are invoked only in the AOR.

Security checking

For function-shipped requests, security checking in the FOR is invoked for the first request that refers to a given file in each unit of work. Thus transaction-level security checks can be performed in the FOR.

For cross-memory requests, security checking is invoked only at CONNECT time. Thus transaction-level security checks cannot be performed in the FOR.

Read request failure

If a read request using function shipping fails, the input area is unchanged.

If a read request using cross-memory services fails, there is a chance that the input area will be altered although no record was retrieved. You should not therefore rely on the input area being unchanged, although you can be sure that the key will not have been changed.

EXEC interface block

You might notice that read requests using cross-memory services return a value in the EIBRESP2 field. Function-shipped read requests do not return a value in the EIBRESP2 field, so your applications should not be dependent on this field being set by read requests.

Key length

For function-shipped requests, you must specify the correct key length either in the remote-file definition in the AOR or explicitly on the file request (to match the key length in the VSAM definition in the FOR). If you do not, the INVREQ condition is returned for any request that accesses the file. This applies to any file, not just one that is defined as a data table.

For cross-memory requests, the key length in the AOR is not used; requests can complete successfully even if the key length is not specified in the AOR, or if the key length specified in the AOR does not match that in the FOR. However, your applications should not depend on this because some of the requests might be function shipped.

Differences between SDT services and VSAM

Because SDT services replace VSAM for many data table requests, you should be aware of the following differences in the way that certain requests are implemented.

Read while updating (different transactions)

In the case of a READ command for a data table record following a READ UPDATE issued for that record by *another* transaction and preceding the associated update request, when SDT services are used, the READ command is processed immediately.

When VSAM is used, the READ command waits until the update request is complete.

Read while updating (same transaction)

In the case of a READ command for a data table record following a READ UPDATE issued for that record by *the same* transaction and preceding the associated update request, when SDT services are used, the READ command is processed immediately.

When VSAM is used, the transaction incurs a deadlock abend AFCG.

Delete during browse

When SDT services are used for a STARTBR or RESETBR command for a data table record, it is possible for the record to be deleted before the associated READNEXT or READPREV command is issued.

When VSAM is used, the record cannot be deleted before the associated READNEXT or READPREV command is issued.

Thus, when SDT services are used, if a STARTBR or RESETBR command is issued with a key other than the special 'last record' key, X'FF....', and the record selected is deleted before the READNEXT command, the READNEXT command reads the succeeding record.

If there is no succeeding record, the ENDFILE condition is returned. If the EQUAL option was used on the STARTBR or RESETBR, the key of the record that is read might not match the key specified.

If a STARTBR or RESETBR command is issued with the special 'last record' key, and the selected record is deleted before the READPREV command, the READPREV command reads the preceding record, or returns the ENDFILE condition if there is none.

Write during browse

When SDT services are used, if a browse reads to the end of a file, raising the ENDFILE condition, and a new record is then inserted beyond the end of the file, a subsequent READNEXT will be able to read the new record.

When VSAM is used, the subsequent READNEXT may not be able to find the new record, but will instead report the ENDFILE condition again.

Delete while updating (same transaction)

When SDT services are used for a DELETE command which specifies RIDFLD for a data table record after a READ UPDATE has been issued for that record by the same transaction and before the associated update request, the DELETE command is processed successfully, and the associated update request receives a NOTFND condition.

Chapter 6. Managing SDT resource definitions

You define a data table in the same way as a CICS file except that you need to specify in addition:

- Which type of data table is to be used
- The maximum number of records that can be held in the data table

  Note: The VSAM KSDS definition supplies the maximum record length and the key length.

You can define a file as a data table by the:

- CEDA DEFINE FILE command (the recommended method, see “Defining an SDT” on page 36)
- DFHFCT macro (see the *CICS Resource Definition Guide*)

Also, you can use:

- The EXEC CICS SET FILE or CEMT SET FILE command to change the data table attributes of an existing file
- The EXEC CICS INQUIRE FILE or CEMT INQUIRE FILE command to check the data table attributes of an existing file

See:

- “Using EXEC CICS commands to manage SDT definitions” on page 39
- “Using CEMT commands to manage SDT definitions” on page 40

Defining an SDT

Figure 6 shows the CEDA panel for the CEDA DEFINE FILE command including the data table parameters.

```
 File         ==> ........
 Group        ==> ........
 DEScription  ==> ................................................
VSAM PARAMETERS
 DSNAme       ==> ........................................
 Password     ==> ........          PASSWORD NOT SPECIFIED
 Lsrpoolid    ==> 01                01-15 | None
 Catname      ==>
 DSNSharing   ==> Noreqs            Noreqs | Allreqs | Modifyreqs
 STRings      ==> 001               1 - 255
 Nsrgroup     ==> ........
 SHr4access   ==> Key               Key | Rba
REMOTE ATTRIBUTES
 REMOTESystem ==> ....
 REMOTEName   ==> ........
 RECORDSize   ==> .....             1 - 32767
 Keylength    ==> ...               1 - 255
INITIAL STATUS
 STAtus       ==> Enabled           Enabled | Disabled | Unenabled
 Opentime     ==> Firstref          Firstref | Startup
BUFFERS
 DAtabuffers  ==>                   2 - 32767
 Indexbuffers ==>                   1 - 32767
DATATABLE PARAMETERS
 Table        ==> No                No | Cics | User
 Maxnumrecs   ==> ........          16 - 16777215
DATA FORMAT
 RECORDFormat ==> V                 V | F
OPERATIONS
 Add          ==> No                No | Yes
 Browse       ==> No                No | Yes
 DELete       ==> No                No | Yes
 REAd         ==> Yes               Yes | No
 Update       ==> No                No | Yes
AUTO JOURNALING
 JOurnal      ==> No                No | 1 - 99
 JNLRead      ==> None              None | Updateonly | Readonly | All
 JNLSYNCRead  ==> No                No | Yes
 JNLUpdate    ==> No                No | Yes
 JNLAdd       ==> None              None | Before | AFter |ALl
 JNLSYNCWrite ==> Yes               Yes | No
RECOVERY PARAMETERS
 RECOVery     ==> None              None | Backoutonly | All
 Fwdrecovlog  ==> No                No | 1-99
```

Figure 6. CEDA DEFINE FILE panel

Full details of how to use the CEDA DEFINE FILE command to define files are given in the *CICS Resource Definition Guide*. Only the parameters that relate to data tables are described in this chapter.

ADD(NO|YES), BROWSE(NO|YES), DELETE(NO|YES), READ(YES|NO), and UPDATE(NO|YES)
Specify which of these file operations can be requested for the data table.

CATNAME(name)
Specify the filename of the catalog in which the source data set resides.

DSNAME(name)
Specify the name of the VSAM KSDS that is to be used as the source data set.

FILE(name)
Specify the name of the file.

For a CMT, this name is used to refer to both the data table and the source data set, which are treated as a single entity by CICS.

For a UMT, this name is used to refer to only the data table.

LSRPOOLID(number|1)
Specify the number of the VSAM local shared resource (LSR) pool that is to be used by the data table. CICS uses the LSR integrity function to prevent concurrent reading and updating of the same record by multiple users, so you must specify an LSRPOOL number, in the range 1 through 15. The default is **LSRPOOLID(1)**.

MAXNUMRECS(value)
Specify the maximum number of records that can be contained in the data table, in the range 16 through 16 777 215.

OPENTIME({FIRSTREF|STARTUP})
Specify when the file is to be opened, either on first reference or immediately after startup by the automatically-initiated transaction CSFU. **OPENTIME(FIRSTREF)** is assumed by default.

Remember that the data table is loaded when the file is opened, so, if you are using the user exit XDTRD, make sure that the user exit is activated before the file is opened (see “Activating user exits” on page 41).

RECORDFORMAT({V|F})
Specify the format of the records in the file—either **RECORDFORMAT(V)** for variable-length records or **RECORDFORMAT(F)** for fixed-length records.

RECORDFORMAT(V) is assumed by default. A UMT must have variable-length records.

RECOVERY({NONE|BACKOUTONLY|ALL})
Specify the type of recovery support that is required for the data table. The default is **RECOVERY(NONE)**.

For a UMT, only dynamic transaction backout is supported by CICS, so **RECOVERY(BACKOUTONLY)** and **RECOVERY(ALL)** have the same meaning.

For a CMT, the **RECOVERY** parameter applies to the source data set; it must be consistent with any other file definition for the same data set.

TABLE({NO|CICS|USER})
Specify **TABLE(CICS)** to define the table as a CMT.

Specify **TABLE(USER)** to define the table as a UMT.

If you specify **TABLE(NO)**, or do not specify the **TABLE** parameter, the file is not defined as a data table.

Example of a CMT definition

The following example shows the definition of a CMT. Only the relevant parameters are shown.

```
 File         ==> APPLE
 Group        ==> FRUIT
 DEScription  ==> ........................................
VSAM PARAMETERS
 DSNAme       ==> CIC01.CICOWN.APPLES
 Password     ==> ........           PASSWORD NOT SPECIFIED
 Lsrpoolid    ==> 2                  1-15 | None
 Catname      ==>
 DSNSharing   ==> Allreqs            Noreqs | Allreqs | Modifyreqs
 STRings      ==> 005                1 - 255
 Nsrgroup     ==> ........
 SHr4access   ==> Key                Key | Rba
INITIAL STATUS
 STAtus       ==> Enabled            Enabled | Disabled | Unenabled
 Opentime     ==> STARTUP            Firstref | Startup
DATATABLE PARAMETERS
 Table        ==> CICS               No | Cics | User
 Maxnumrecs   ==> 1000000            16 - 16777215
DATA FORMAT
 RECORDFormat ==> F                  V | F
OPERATIONS
 Add          ==> YES                No | Yes
 Browse       ==> No                 No | Yes
 DELete       ==> YES                No | Yes
 REAd         ==> YES                Yes | No
 Update       ==> YES                No | Yes
RECOVERY PARAMETERS
 RECOVery     ==> ALL                None | Backoutonly | All
 Fwdrecovlog  ==> 10                 No | 1-99
```

Figure 7. Defining a CICS-maintained data table

Example of a UMT definition

The following example shows the definition of a UMT. Only the relevant parameters are shown.

```
 File         ==> COURGET
 Group        ==> VEGS
 DEScription  ==> ........................................
VSAM PARAMETERS
 DSNAme       ==> CIC02.CICOWN.COURGET
 Password     ==> ........           PASSWORD NOT SPECIFIED
 Lsrpoolid    ==> 5                  1-15 | None
 Catname      ==>
 DSNSharing   ==> Allreqs            Noreqs | Allreqs | Modifyreqs
 STRings      ==> 005                1 - 255
 Nsrgroup     ==> ........
 SHr4access   ==> Key                Key | Rba
INITIAL STATUS
 STAtus       ==> Enabled            Enabled | Disabled | Unenabled
 Opentime     ==> FIRSTREF           Firstref | Startup
DATATABLE PARAMETERS
 Table        ==> USER               No | Cics | User
 Maxnumrecs   ==> 200000             16 - 16777215
DATA FORMAT
 RECORDFormat ==> V                  V | F
OPERATIONS
 Add          ==> YES                No | Yes
 Browse       ==> YES                No | Yes
 DELete       ==> No                 No | Yes
 REAd         ==> YES                Yes | No
 Update       ==> YES                No | Yes
RECOVERY PARAMETERS
 RECOVery     ==> BACKOUTONLY        None | Backoutonly | All
 Fwdrecovlog  ==> No                 No | 1-99
```

Figure 8. Defining a user-maintained data table

Using EXEC CICS commands to manage SDT definitions

This section contains General-use Programming Interface and Associated Guidance Information.

You can use the EXEC CICS SET FILE command to change the definition of an existing SDT, and the EXEC CICS INQUIRE FILE command to check the definition of an existing SDT. For programming information, including details of how to use these commands and the parameters described here, see the *CICS System Programming Reference* manual. The parameters that are relevant to data tables are described below.

EXEC CICS SET FILE command

The following parameters are relevant to data tables; you can use them only when the file is closed and disabled. You can specify a data table attribute of a file in a CICS-value data area (cvda):

TABLE(cvda)

Specify a cvda value of **CICSTABLE** to define the file as a CMT.

Specify a cvda value of **USERTABLE** to define the file as a UMT.

Specify a cvda value of **NOTTABLE** to indicate that the file is not a data table.

EXEC CICS INQUIRE FILE command

The following parameters are relevant to data tables. You can request that each data table attribute of a file is returned in a CICS-value data area (cvda) by specifying:

TABLE(cvda)

If the value **CICSTABLE** is returned, the file has been defined as a CMT.

If the value **USERTABLE** is returned, the file has been defined as a UMT.

If the value **NOTTABLE** is returned, the file is not currently defined as a data table.

If the value **NOTAPPLIC** is returned, the option is not applicable because the file is a remote file.

Using CEMT commands to manage SDT definitions

You can use the CEMT SET FILE command to change the definition of an existing SDF, and the CEMT INQUIRE FILE command to check the definition of an existing SDF. Full details of how to use these commands, including the parameters described here, are given in *CICS-Supplied Transactions* manual. The parameters that are relevant to data tables are described below.

CEMT SET FILE command

The following parameters are relevant to data tables; you can use them only when the file is closed and disabled.

{CICSTABLE|USERTABLE|NOTTABLE}

Specify **CICSTABLE** to define the file as a CMT.

Specify **USERTABLE** to define the file as a UMT.

Specify **NOTTABLE** to indicate that the file is not a data table.

MAXNUMRECS(value)

Specify the maximum number of records that can be contained in the data table, in the range 16 through 16 777 215.

CEMT INQUIRE FILE command

The following parameters are relevant to data tables.

Data table

If the value **CICSTABLE** is returned, the file has been defined as a CMT.

If the value **USERTABLE** is returned, the file has been defined as a UMT.

If the value **NOTTABLE** is returned, the file is not currently defined as a data table.

MAXNUMRECS(value)

The value returned indicates the maximum number of records that can be contained in the data table. The value is zero if no value has been set previously.

Chapter 7. Using the CICS-supplied global user exits

This chapter contains Product-sensitive Programming Interface and Associated Guidance Information.

This chapter describes the three global user exit points that are included in data table services. You can supply one or more assembler-language programs to be executed at each of these points in order to extend or modify the function provided by CICS.

The chapter is divided into:

- Activating user exits, which provides information on activating user exits.
- "Communicating between CICS and exit programs" on page 42, which provides general information.
- "XDTRD user exit" on page 44. XDTRD is invoked for each record that is read from the source data set (normally when the file is being loaded). You can then select whether to load the record into the data table or not. For a UMT, you can also modify the record.
- "XDTAD user exit" on page 45. XDTAD is invoked for each record that is added to the source data set. You can then select whether to add the record to the data table or not.
- "XDTLC user exit" on page 46. XDTLC is invoked when loading of the data table is complete, whether successful or not.

Activating user exits

To activate the data table user exits, you need to perform the following steps:

1. Decide which user exits you want to use. A description of each user exit is included in this chapter.
2. Write the user exit programs. Examples are included in Appendix A, "Sample user exit programs" on page 69.
3. Define the user exit programs to CICS, using the CEDA DEFINE PROGRAM command (see the *CICS Resource Definition Guide*).
4. Activate the user exits, using the EXEC CICS ENABLE command (for programming information about this command, see the *CICS System Programming Reference* manual). If required, you can later deactivate the user exits, using the EXEC CICS DISABLE command.

Unless you control the opening of a data table explicitly, with a CEMT or EXEC CICS command, you should probably activate the user exits during CICS startup. Otherwise loading of the data table might begin before the user exits are activated. To activate the user exits during startup, you need to:

1. Write one or more program list table postinitialization (PLTPI) programs that include the EXEC CICS ENABLE commands to activate the user exits (for programming information about PLTPI programs, see the *CICS Customization Guide*).

2. Define a program list table (PLT) with an entry for each of those PLTPI programs, as described in the *CICS Resource Definition Guide*.
3. Specify the **PLTPI=suffix** parameter for system initialization, as described in the *CICS System Definition Guide*. Use the suffix of the PLT that was defined in the previous step. This causes the PLTPI programs to be executed in the second stage of initialization, before any files are opened.

You can use PLT shutdown (PLTSD) programs in a similar way to disable the user exits during CICS shutdown.

In addition:

- Samples of exit programs are shown in Appendix A, “Sample user exit programs” on page 69.
- Programming information about global user exits and how to use them is given in the *CICS Customization Guide*.

The EXEC interface user exits XEIIN and XEIOUT and the file control user exits XFCREQ and XFCREQC are not invoked in the file-owning region if a request to access a data table is satisfied by cross-memory services.

Communicating between CICS and exit programs

A parameter list is used to pass information between CICS and the data table exit programs. CICS Transaction Server for VSE/ESA Release 1 supplies a copybook, named DFHXDTDS, which contains a DSECT to define this parameter list. You should include a COPY DFHXDTDS statement in each of your exit programs. The DSECT is shown in Figure 9 on page 43.

The field names used in this DSECT are referenced in the user exit descriptions that follow the figure.

```
*************************************************************************
*                                                                       *
*   Data Table Parameter List for User Exits XDTRD, XDTAD and XDTLC.    *
*                                                                       *
*   Some of the parameters are only used by one or two of the exits.    *
*   This is indicated in the comments for those parameters.             *
*   The comments also indicate whether the field is used for input      *
*   (In), output (Out), or both (In/Out).                               *
*                                                                       *
*   NOTE that this definition could be used by exit programs running    *
*   both on CICS regions which have shared data tables support          *
*   (SDT) installed and on ones which do not, providing the UEPDTSDT    *
*   flag is used to test whether SDT is installed, and that the         *
*   parameters which are unique to SDT are only used when it is set.    *
*                                                                       *
*************************************************************************
DT_UE_PLIST_DSECT  DSECT ,
DT_UE_PLIST        DS  0XL84           Data Table User Exits          X
                                       Parameter List
UEPDTNAM             DS  CL8           Data table name (In)
UEPDTFLG             DS  0CL1          Flags (In):
                     DS  BL1
UEPDTSDT               EQU X'80'       Exit invoked by SDT support
*-----------------------------------------------------------------------*
*           The remaining flags are available only to exits which       *
*           have been invoked by shared data tables support             *
*-----------------------------------------------------------------------*
UEPDTCMT               EQU X'40'       Table is CICS-maintained
UEPDTOPT               EQU X'20'       Exit invoked by table loader,  X
                                       so optimization by skipping    X
                                       may be requested - XDTRD only
*                      EQU X'1F'       Reserved
UEPDTORC             DS  AL1           Data table load return code -  X
                                       XDTLC only, values below (In)
                     DS  BL2           Reserved
```

Figure 9 (Part 1 of 2). Data table user exit parameter list

```
UEPDTRA             DS  A             Data record address - XDTRD    X
                                      and XDTAD only (In)
UEPDTRBL            DS  F             Data buffer length - XDTRD and X
                                      XDTAD only (In)
UEPDTRL             DS  F             Data table record length -     X
                                      XDTRD and XDTAD only, XDTRD    X
                                      can return new length in here  X
                                      if it amends record (only      X
                                      allowed for UMT) (In/Out)
UEPDTKA             DS  A             Key address - XDTRD and XDTAD  X
                                      only (In)
UEPDTKL             DS  F             Key length - XDTRD and XDTAD   X
                                      only (In)
*---------------------------------------------------------------------*
*         The following fields are only available to exits which      *
*         have been invoked by shared data tables support             *
*---------------------------------------------------------------------*
UEPDTDSL            DS  F             Length of data set name (In)
UEPDTDSN            DS  CL44          Source data set name (In)
UEPDTSKA            DS  A             Address of skip-key area: exit X
                                      should return a key of length  X
                                      UEPDTKL in this area if it has X
                                      requested optimization of load X
                                      by skipping - XDTRD only (In)
*---------------------------------------------------------------------*
*   Values for UEPDTORC (supplied to XDTLC exit only)                 *
*---------------------------------------------------------------------*
UEPDTLCS          EQU 0               load completed successfully
UEPDTLFL          EQU 128             load failed
```

Figure 9 (Part 2 of 2). Data table user exit parameter list

The user exits should set a return code in register 15. The return code values are supplied by the DFHUEXIT macro. The valid values for each user exit are given in the following descriptions.

If you want your exit programs to work for both basic and shared data tables support, you can check UEPDTFLG to find out which version of data tables support invoked the exit program. For SDT, this flag byte also indicates which type of data table is being used and whether the exit program is being invoked during loading.

The exit program should use either the filename (field UEPDTNAM) or the name of the source data set (see fields UEPDTDSN and UEPDTDSL) to determine whether this is a file for which any action is to be taken.

You can enable several exit programs at the same exit point, each of which, for example, takes action for a particular file or data set.

XDTRD user exit

The XDTRD user exit is invoked just before CICS attempts to add to the data table a record that has been retrieved from the source data set.

This normally occurs when the loading process retrieves a record during the sequential copying of the source data set. However, it can also occur when an application retrieves a record that is not in the data table and:

- For a user-maintained data table, loading is still in progress, or
- For a CICS-maintained data table, loading terminated before the end of the source data set was reached (because, for example, the data table was full).

The record retrieved from the source data set is passed as a parameter to the user exit program—see fields UEPDTRA and UEPDTRL. This program can choose (depending, for example, on the key value—see fields UEPDTKA and UEPDTKL) whether to include the record in the data table or not.

Alternatively, the exit program can request that all subsequent records up to a specified key are skipped—see field UEPDTSKA; these records are not passed to the exit program. This facility is available only during loading. You can specify the key as a complete key, or you can specify just the leading characters by padding the skip-key area with binary zeros.

The action required is indicated by setting the return code. Depending on the return code value, the following action is taken by CICS:

Table 5. Return codes for XDTRD user exit. A value of UERCPURG should be returned if the exit program has received a PURGED response to a call that it has issued.

Return code	Action
UERCDTAC	Include the record in the data table. This is the default if the exit is not activated.
UERCDTRJ	Do not include the record in the data table.
UERCDTOP	Skip over this record and the following records until a key is found that is equal to or greater than the key specified in the skip-key area.

For a UMT, the program can also modify the data in the record to reduce the storage needed for the data table. Application programs that use the data table must be aware of any changes made to the record format by the exit program. If the record length is changed, the exit program must set the new length in the parameter list—see field UEPDTRL. The new length must not exceed the data buffer length—see field UEPDTRBL.

XDTAD user exit

The XDTAD user exit is invoked when a write request is issued to a data table.

- For a UMT, the user exit is invoked once—before the record is added to the data table.
- For a CMT, the user exit is invoked twice—before the record is added to the source data set and then again before the record is added to the data table.

The record written by the application is passed as a parameter to the user exit program—see fields UEPDTRA and UEPDTRL. This program can choose (depending on the key value, for example—see fields UEPDTKA and UEPDTKL) whether to include the record in the data table or not. This decision is indicated by setting the return code.

Depending on the return code value, the following action is taken by CICS:

Table 6. Return codes for XDTAD user exit. A value of UERCPURG should be returned if the exit program has received a PURGED response to a call that it has issued.

Return code	Action
UERCDTAC	Add the record to the data table. This is the default if the exit is not activated.
UERCDTRJ	Do not add the record to the data table.

The XDTAD exit must not modify the data in the record. If you used XDTRD to truncate the data records when the user-maintained data table was loaded, you must code your application so that it only tries to write records of the correct format for the data table.

XDTLC user exit

The XDTLC user exit is invoked at the completion of data table loading—whether successful or not. **The user exit is not invoked if the data table is closed for any reason before loading is complete.**

The exit program is informed if the loading did not complete successfully—see field UEPDTORC. This could occur, for example, if the maximum number of records was reached or there was insufficient virtual storage. In this case, the exit program can request that the file is closed immediately, by setting the return code.

Depending on the return code value, the following action is taken by CICS:

Table 7. Return codes for XDTLC user exit. A value of UERCPURG should be returned if the exit program has received a PURGED response to a call that it has issued.

Return code	Action
UERCDTOK	No action; the file remains open. This is the default if the exit is not activated.
UERCDTCL	Close the file.

Chapter 8. Using shared data tables services

This chapter describes these operational aspects of data tables:

Opening a data table

A data table must be opened before it can be used by an application. This is done in the same way as for any CICS file, by one of the following methods:

- Automatically, by the CICS-supplied transaction CSFU, at the end of CICS startup, if the data table is defined with OPENTIME(STARTUP) or FILSTAT=OPENED.
- Explicitly, by a CEMT or EXEC CICS request issued by the user.
- Implicitly, on first reference to the data table, if the data table is defined with OPENTIME(FIRSTREF) or FILSTAT=CLOSED. The first remote access to a closed data table implicitly opens it.

All the rules and options for opening a CICS file also apply to a file that is defined as a data table. In addition, the loading of the data table is initiated.

For a large data table, loading could take a significant time. Chapter 5, "Application programming for shared data tables" on page 27 discusses the application programming commands that can be used with a UMT, and the way that performance gains that can be achieved with a CMT are limited until loading is completed.

During the opening of the file:

1. The access method control block (ACB) for the VSAM source data set is opened under a separate VSE subtask. This step is the same as for any CICS file.
2. For the first data table used by a region, CICS:
   - Creates VSE storage subpools for use by SDT
   - Creates a VSE/ESA data space for use by this region's data tables
   - Attempts a LOGON operation as a server
3. A special CICS transaction, CSSY, is attached to load the data table into the data space.
4. The transaction that issued the request to open the data table can now continue processing.
5. CICS issues a message DFHFC0940 to indicate that loading has started. The message is sent to the CSFL transient data queue.
6. The transaction that loads the data table reads the source data set sequentially. Under the optional control of the user exit XDTRD, the transaction copies the records into the data space. It also constructs an index in address-space storage to facilitate subsequent access to the records.

7. CICS issues a message to indicate the result of the loading. The message number is:
   - If loading is successful—DFHFC0941
   - If loading fails—DFHFC0942, DFHFC0943, DFHFC0945, DFHFC0946, DFHFC0947, or DFHFC0948.

   The message is sent to the CSFL transient data queue. Also, if loading fails, the message is sent to the console. For descriptions of these messages, see the *VSE/ESA Messages and Codes Volume 3* manual.
8. When loading is complete (whether successful or not), the user exit XDTLC is invoked if it is active. If the loading was not completed successfully, the exit program can request that the data table is closed.
9. For a UMT, the ACB for the source data set is closed when loading is complete. The data set is deallocated if it was originally dynamically allocated and no other ACBs are open for it.

Note: During an emergency restart, any file that requires backout action is reopened. However, if the file is defined as a data table, loading is not initiated at that time; instead, it is initiated by the CSFU transaction at the end of the emergency restart. This gives an opportunity for any user exits that control the copying of records to the data table during loading to be activated at any stage of PLTPI processing.

Closing a data table

A data table is closed in the same way as for any CICS file, by one of the following methods:

- Explicitly, by a CEMT or EXEC CICS request issued by the user
- Implicitly, when CICS is shutdown normally

All the rules and options for closing a CICS file also apply to a data table. In particular:

- The rules about the quiescing of current users of the file apply (except that the file can be closed even when a transaction that is running in an AOR is in the middle of a browse sequence).
- If a UMT is defined as recoverable, all units of work that have changed the data table must complete before the data table can be closed.

The data space storage that is used for the data table records, and the address-space storage that is used for the associated table-entry and index storage is freed as part of the close operation. If a file is reopened after it has been closed, the processing is the same as if the file had not been previously opened.

Interpreting data table statistics

This section describes the statistics information that is produced by CICS to help you monitor the activity on your data tables.

You can use the information contained in a CICS statistics report to evaluate the benefits of using data table services. The information that is recorded for a data table is shown in Table 8 on page 49.

Table 8. Data tables statistics	
Heading	**Description**
Close Type	Type of data table close (only appears in the statistics collected when a data table is closed).
Read Requests	Number of attempts to read records from data table or, in the AOR statistics, the number of read requests that had to be function shipped.
Recs ¬ in Table	Number of attempts to read records from source data set because the record was not found in data table. (For a user-maintained data table, this happens only during loading.)
Adds from Reads	Number of attempts to copy records from source data set to data table during loading process (including read requests from applications during loading).
Add Requests	Number of attempts to write records to data table.
Adds Rejected - Exit	Number of records suppressed by XDTRD and XDTAD user exits.
Adds Rejected - Table Full	Number of records that were not included because the table was full.
Rewrite Requests	Number of attempts to rewrite records in data table.
Delete Requests	Number of attempts to delete records from data table.
Highest Table Size	Peak number of records held in data table.
Storage Alloc(K)	Number of KB allocated to data table.

The statistics for data tables are included in the FILES section of the statistics report. Four sections of FILE information are produced:

- The “FILES - Resource Information” shows information such as the filename, source data set name, data set type (which is always `K` for a data table, because the source must be a VSAM KSDS) and a DT indicator that is explained below.
- The “FILES - Requests Information” shows the statistics for accesses to the source data set. For a loaded UMT, this contains only zeros.
- The “FILES - Data Table Requests Information” shows statistics for accesses to the data table. The meanings of the column headings are given in Table 8.
- The “FILES - Performance Information” shows the use of VSAM strings and buffers, and is only of interest for a data table in that it relates to the source data set.

A request to a data table that is owned by another region is recorded in the statistics report for both the requesting and file-owning regions.

The `DT Indicator` is a single character that can have the values:

T The statistics report contains data table information because the file has been opened as a data table.

R The statistics report contains information for a remote file that has accessed a data table using shared data tables cross-memory services.

S The statistics report contains data table information for the accesses made by this file to an associated data table (which has the same source data set).

X The statistics report contains information for an alternate index file, and reports data table information for the update of the data table associated with a file in the same upgrade set.

The field is blank if data table statistics are not present for the file.

The `Close Type` is a single character that appears in the statistics only for the closing of a data table. The possible values indicate the type of close:

C Close of a CMT.

P Partial close of a CMT. There are still other files using the data table, so the data table itself has not been closed, only the file.

S Close of the source data set for a UMT. This happens at the end of loading of the data table.

U Close of a UMT.

A total value for the storage allocated is not included in the TOTALS line.

To find out how to obtain a statistics report, using the statistics utility program DFHSTUP, see the *CICS Operations and Utilities Guide*. For a description of the data contained in a statistics report, see the *CICS Performance Guide*.

Sample data table statistics

Figure 10 on page 51, Figure 11 on page 52, and Figure 12 on page 53 show extracts from the file statistics for a hypothetical configuration similar to that discussed in "Using statistics to select data tables" on page 18, following conversion of the files into data tables. These figures are used to discuss how to interpret the information about data tables which is provided by the CICS statistics.

```
Requested Statistics Report        Collection Date-Time 12/25/99-23:55:41  Last Reset 21:00:00  Applid CICFOR    Jobname SDTGSTF1

FILES - Resource Information

  File      Dataset Name                                Dataset    DT      Time     Time     Remote    Remote   Lsrpool
  Name           Base Dataset Name (If Applicable)       Type   Indicator Opened   Closed    Name     Sysid     ID

  APPLE     CIC01.CICOWN.APPLES                          K         T    22:12:04  OPEN                          1
  BANANA    CIC01.CICOWN.BANANAS                         K         T    22:53:56  OPEN                          1
  ORANGE    CIC01.CICOWN.CITRUS                          K         T    20:51:25  OPEN                          2
  PLUM      CIC01.CICOWN.PLUMS                           K         T    21:30:10  OPEN                          4
  POTATO    CIC01.CICOWN.POTATOES                        K         T    20:30:10  OPEN                          8
  VICTORI   CIC01.CICOWN.PLUMS                           K         S    21:56:23  OPEN                          4

Requested Statistics Report        Collection Date-Time 12/25/99-23:55:41  Last Reset 21:00:00  Applid CICFOR    Jobname SDTGSTF1

FILES - Requests Information

  File      Get       Get Upd  Browse   Update   Add      Delete   VSAM EXCP Requests
  Name      Requests  Requests Requests Requests Requests Requests   Data    Index

  APPLE          1       495   760001      495       12         0    14560     1321
  BANANA         0      1803    48603     1803      951         0     8212      481
  ORANGE      1087    102677      104   107897     4709      1880    25180     1947
  PLUM           0        10     2001        0       20        10      580        3
  POTATO         0         0    24173        0        0         0     4513        2
  VICTORI        0         3        0        0        5         3        5        1
  *TOTALS*    1088    104988   834882   110195     5697      1893

Requested Statistics Report        Collection Date-Time 12/25/99-23:55:41  Last Reset 21:00:00  Applid CICFOR    Jobname SDTGSTF1

FILES - Data Table Requests Information

  File      Close    Read       Recs ¬   Adds from Add      Adds rejected Adds rejected Rewrite  Delete   Highest    Storage
  Name      Type     Requests   in Table   Reads   Requests    - Exit     - Table Full  Requests Requests Table Size Alloc(K)

  APPLE             1105241          1   760000        12          0             0        495        0   760012      95104
  BANANA             652195          0    48602       951          2             0       1803        0    49551      11200
  ORANGE               1473       1191        0      4709          0             0     107897     1880   604167     266658
  PLUM                  227          1     2000        20          0             0          0       10     2025        256
  POTATO               9670      24173    24165         0          0         24165          0        0     1000       1760
  VICTORI              3063          1        0         5          0             0          0        3     2025        256
  *TOTALS*          1771869      25367   834767      5697          2         24165     110195     1893   760012
```

Figure 10. CICFOR requested file statistics

```
Requested Statistics Report       Collection Date-Time 12/25/99-23:56:58  Last Reset 21:00:00  Applid CICAOR1   Jobname SDTGSTA1

FILES - Resource Information

File      Dataset Name                              Dataset    DT      Time     Time     Remote    Remote   Lsrpool
Name           Base Dataset Name (If Applicable)     Type   Indicator Opened   Closed    Name     Sysid     ID

APPLE     REMOTE                                             R     CLOSED   CLOSED   APPLE     CIF1       N
BANANA    REMOTE                                             R     CLOSED   CLOSED   BANANA    CIF1       N
ORANGE    REMOTE                                             R     CLOSED   CLOSED   ORANGE    CIF1       N
POTATO    REMOTE                                             R     CLOSED   CLOSED   POTATO    CIF1       N
ZUCCINI   REMOTE                                             R     CLOSED   CLOSED   COURGET   CIA2       N

Requested Statistics Report       Collection Date-Time 12/25/99-23:56:58  Last Reset 21:00:00  Applid CICAOR1   Jobname SDTGSTA1

FILES - Requests Information

File      Get      Get Upd  Browse   Update   Add      Delete   VSAM EXCP Requests
Name      Requests Requests Requests Requests Requests Requests   Data    Index

APPLE          1      520        0      520        5        0        0        0
BANANA         0        0        0        0        0        0        0        0
ORANGE       214    38735       14    37105     1311      630        0        0
POTATO         0        0    24173        0        0        0        0        0
ZUCCINI        0        0        0        0        0        0        0        0

*TOTALS*     215    39255    24187    38625     1316      630

Requested Statistics Report       Collection Date-Time 12/25/99-23:56:58  Last Reset 21:00:00  Applid CICAOR1   Jobname SDTGSTA1

FILES - Data Table Requests Information

File      Close    Read       Recs ¬  Adds from Add      Adds rejected Adds rejected Rewrite  Delete   Highest    Storage
Name      Type     Requests   in Table  Reads   Requests    - Exit     - Table Full  Requests Requests Table Size Alloc(K)

APPLE             1304214          0        0        0          0             0          0        0          0
BANANA             441349          0        0        0          0             0          0        0          0
ORANGE              63384        228        0        0          0             0          0        0          0
POTATO               4835      24173        0        0          0             0          0        0          0
ZUCCINI             97867          0        0        0          0             0          0        0          0

*TOTALS*          1911649      24401        0        0          0             0          0        0          0
```

Figure 11. CICAOR1 requested file statistics

```
Requested Statistics Report       Collection Date-Time 12/25/99-23:59:59  Last Reset 21:00:00  Applid CICAOR2   Jobname SDTGSTA2
FILES - Resource Information
  File      Dataset Name                              Dataset    DT      Time     Time     Remote    Remote   Lsrpool
  Name           Base Dataset Name (If Applicable)     Type   Indicator Opened   Closed     Name     Sysid      ID

  COURGET   CIC02.CICOWN.COURGET                        K         T    22:35:02  OPEN                          1
  LEMON     REMOTE                                               ?     CLOSED   CLOSED   ORANGE    CIF1        N

Requested Statistics Report       Collection Date-Time 12/25/99-23:59:59  Last Reset 21:00:00  Applid CICAOR2   Jobname SDTGSTA2
FILES - Requests Information
  File      Get      Get Upd  Browse   Update   Add      Delete   VSAM EXCP Requests
  Name      Requests Requests Requests Requests Requests Requests   Data    Index

  COURGET          0        0        0        0        0        0        0        0
  LEMON          946    63942      299    70792     3398     1250        0        0
  *TOTALS*       946    63942      299    70792     3398     1250

Requested Statistics Report       Collection Date-Time 12/25/99-23:59:59  Last Reset 21:00:00  Applid CICAOR2   Jobname SDTGSTA2
FILES - Data Table Requests Information
  File      Close    Read       Recs ¬   Adds from Add      Adds rejected Adds rejected Rewrite  Delete   Highest    Storage
  Name      Type     Requests   in Table   Reads   Requests    - Exit     - Table Full  Requests Requests Table Size Alloc(K)

  COURGET              27656          0        0    386590          0             0       27656        0   101232      16160
  LEMON              3240872       1245        0         0          0             0           0        0        0
  *TOTALS*           3268528       1245        0    386590          0             0       27656        0   101232
```

Figure 12. CICAOR2 requested file statistics

The main changes that are seen in the statistics when a file is redefined as a data table are:

1. Data table statistics appear in both the owning and requesting regions showing the use of the data table.
2. Values appear in the owning region showing the activity of loading the data table.
3. Except during loading, the counts of `Get Requests` and `Browse Requests` in the "FILES - Requests Information" statistics are much reduced (often to zero) because such requests can now be satisfied from the data table.
4. The `Recs ¬ in Table` figure indicates the degree to which benefit from shared data tables support has been prevented. In an AOR it shows how many retrieval requests have had to be function shipped. In an FOR it shows how many records have had to be fetched from the source data set.

You should use the statistics to get an overall feel for the behavior of your data tables, rather than attempt to explain the individual values.

The examples demonstrate a number of points about the statistics. These points are discussed in the rest of this section.

Normal loading

`Adds from Reads` usually shows the number of attempts to add a record during loading[1]. Because the loading process involves browsing the source data set until the end of the file, the number of `Browse Requests` (in the "FILES - Requests Information") equals `Adds from Reads` + 1 if loading completed successfully and there have been no other browses on the source data set. The statistics for APPLE in Figure 10 on page 51 illustrate this point.

In the CICFOR statistics shown, APPLE, BANANA, and PLUM were opened after the last statistics reset, but ORANGE and POTATO were opened before, so the latter do not display the load-time statistics. It is generally better to assess statistics from a time interval that has not been distorted by loading, but you should remember that the loading process incurs an overhead that has to be recovered by the number of data table accesses.

Optimization of loading

The statistics for BANANA in Figure 10 on page 51 show an example in which only a range of key values from the middle of the source data set is required in the data table. Here, the XDTRD exit has been used to skip over any keys that are not in this range. For information about the exit, see "XDTRD user exit" on page 44.

`Adds rejected - Exit` shows the number of times the exit returned a non-zero return code, and is 2 in this case: 1 for when the first record in the source data set was presented, and the exit requested the load to skip on to the first key in the desired range, and 1 for when the first key beyond this range was presented, and the exit requested the load to skip over all remaining records to the end of the source data set.

In a case like this, you would usually use the XDTAD exit to reject any records that are written with keys outside the desired range. Then, the number of `Adds rejected - Exit` would include the number of such records that had been written to the file.

The number of `Adds from Reads` contains the number of records that were loaded into the data table plus the two that were rejected. As for all the file and data table statistics, this figure shows the number of attempted, rather than the number of successful, writes.

Loading a UMT

When the loading of a UMT completes, the source data set is closed and an unsolicited statistics record is written that reports the number of records that are browsed from the source and written to the data table (or rejected by the XDTRD exit). Therefore, these figures do not appear in any later statistics reports, such as that for COURGET in Figure 12 on page 53. A `Close Type` of `S` identifies such statistics.

---

1 If loading has completed, but the load failed to read to the end of the source data set, the count for a CMT might also show attempts to add records that have been read from the source data set because they were not originally loaded.

Implicit open from the requesting region

If the file is not open in the FOR when the AOR issues the first read to it, then no connection exists and the read is function shipped. This appears as one file `Get Request` in the AOR statistics. The implicit open and subsequent loading of the data table is triggered in the FOR.

This first read attempt from the new data table is counted in the `Read Requests` in the FOR data table statistics, but as the record is not found in the data table at this stage, it is added to the count of `Recs ¬ in Table` (which records the number of times a record could not be obtained from the data table). The record is fetched from VSAM, so the number of file `Get Requests` is incremented by 1. The statistics from CICAOR1 and CICFOR for APPLE illustrate this point.

Update requests

All update requests (writes, rewrites, and deletes) are processed by the owning region, which also controls loading and other maintenance of the data table. Because of this, the data table statistics of `Adds from Reads`, `Add Requests`, `Adds rejected`, `Rewrite Requests`, `Delete Requests`, `Highest Table Size`, and `Storage Alloc` are always zero on the remote requesting regions.

Updates are always reflected in both the data table and the source data set for a CMT, so matching numbers are often seen in the file statistics and the data table statistics for `Add Requests` and `Delete Requests`, and also for `Update Requests` in the file statistics compared with `Rewrite Requests` in the data table statistics. The statistics for APPLE and ORANGE illustrate this point.

These numbers might not match if not all records in the source data set are loaded into the data table, or if some error occurs when the source data set is updated. For example, an attempt to write a record with a duplicate key to the source data set is counted in the file `Add Requests` but no attempt is made to write the record to the data table, so the count of data table `Add Requests` is less.

In the case of a UMT, access after loading is complete is always to the data table, so the line of file statistics always contains zeros. The statistics for COURGET in Figure 12 on page 53 illustrate this point.

Data table high water mark

The `Highest Table Size` shows the largest number of records that was present in the data table at any one time. For APPLE, from which no records have been deleted, this is the number of records originally loaded, plus the number of data table `Add Requests`. For BANANA, the XDTRD user exit accepts only records that are of interest during loading, and XDTAD performs the same task when records are written; so the number is given by `Adds from Reads` plus `Add Requests` minus `Adds rejected - Exit`.

Total storage allocated

`Storage Alloc` gives the number of Kilobytes that have been allocated to the data table. This includes both address space and data space storage.

Reading and browsing

The number of data table Read Requests includes browses that are satisfied by the data table. Thus for ORANGE in Figure 11 on page 52 and LEMON in Figure 12 on page 53 the numbers of file Browse Requests are very small (and in most cases they would be zero). But the number of data table Read Requests is of a similar magnitude to the total number of Get and Browse requests that were made before conversion of the file to a data table.

Failure to access records using the data table

The set of figures for ORANGE and LEMON show an effect that is sometimes seen when there is much update activity on a CMT. In this case, some of the read requests from remote regions might find that the record in the data table is being updated, so these requests are function shipped to the FOR.

For example, LEMON shows 1245 Recs ¬ in Table, of which 946 are Get Requests and 299 are Browse Requests. The function-shipped reads and browses attempt to access the data table in CICFOR (as shown by the 1473 Read Requests for ORANGE), by which time some of the reads can be satisfied from the data table, but the remainder use the source data set (as shown by the number under Recs ¬ in Table).

POTATO shows what can happen if an unsuitable choice of candidate is made. Because the data table size specified in the file definition is much less than the number of records in the source data set, only a small part of the file is loaded. However, the file is accessed remotely by an application that browses many records that lie beyond those that were loaded.

All those browse requests have to be function shipped to CICFOR (as shown by the high number of Recs ¬ in Table seen in Figure 11 on page 52 for that file) and then, on CICFOR, the source data set has to be accessed (as shown by the number of Browse Requests in Figure 10 on page 51). An attempt is made to add the records to the data table (see count of Adds from Reads) but as the data table is still at its maximum number of records, they have to be rejected (see Adds rejected - Table Full). Incidentally, the statistics record that contains the loading figures for the file includes 1 under Adds rejected - Table Full for the record that caused the load to terminate because the data table had reached its maximum size.

The values of Read Requests, Recs ¬ in Table, Browse Requests, Adds from Reads, and Adds rejected - Table Full are not all equal (as might have been expected) because some browses reach the end of the source data set (in which case there is no record to attempt to add to the data table) and also because often no attempt is made to access the data table when browsing over a range of records that is known to be missing from the data table.

Although this is an extreme example, it does illustrate the importance in certain situations of having a good understanding of the applications. A user exit should have been used to select the range of records on which most of the browsing occurs.

Multiple files with a single source

The file VICTORI in CICFOR is included to show that when a second file is defined with the same source data set as a file that is open as a data table (PLUM) then it can take advantage of the data table for non-update reads and browses (regardless of which file is opened first). In the Resource Information for VICTORI, a `DT Indicator` of `S` means that the line of data table statistics shows how the data table has been used by this associated file; for example, 3063 reads or browses have been satisfied from the data table.

The same `Storage Alloc` and `Highest Table Size` statistics are reported for both PLUM and VICTORI because the data table is associated with both files. Because of this, the data table `TOTALS` line does not include a value for the total storage allocated. The load-time statistics are reported only for the file that initiated the data table; that is, the file whose open caused the current instance of the data table to be built.

Additional statistics fields

Some additional statistics about shared data tables are collected when file statistics are gathered, but they are not formatted in a statistics report. You can write a program to extract these additional statistics from the statistics record. For programming information about CICS statistics, see the *CICS Customization Guide*.

This section describes the fields that are related to shared data tables and are part of the DFHA17 statistics record but are not displayed in a statistics report.

A17_DT_SIZE_CURRENT: a fullword at offset 160 (X'A0') containing the current count of records in the data table.

A17_DT_IN_USE_TOTAL a fullword at offset 168 (X'A8') containing the total amount of storage (in KB) currently in use for the data table.

A17_DT_ALLOC_ENTRY a fullword at offset 172 (X'AC') containing the amount of storage (in KB) currently allocated from the server's address space to hold table entries for this data table.

A17_DT_IN_USE_ENTRY a fullword at offset 176 (X'B0') containing the amount of storage (in KB) in the server's address space currently being used by table entries for this data table.

A17_DT_ALLOC_INDEX a fullword at offset 180 (X'B4') containing the amount of storage (in KB) currently allocated from the server's address space to the index for this data table.

A17_DT_IN_USE_INDEX a fullword at offset 184 (X'B8') containing the amount of storage (in KB) in the server's address space currently being used by the index for this data table.

A17_DT_ALLOC_DATA a fullword at offset 188 (X'BC') containing the amount of storage (in KB) currently allocated from the data space for the data portion of the records in this data table.

A17_DT_IN_USE_DATA a fullword at offset 192 (X'C0') containing the amount of storage (in KB) in the data space currently being used to hold record data for this data table.

A17_DT_REREADS a fullword at offset 196 (X'C4') containing a count of the number of times a read from a requesting region has retried a part of the data table section of the request processing because the data table changed in some way after the start of that section.

These fields are also displayed in the X'0B22' exit trace for a statistics call amongst the fourteen statistics fullwords, which are (in the order they appear in the trace):

```
Adds from Reads  during load
Adds rejected - Exit  during load
Adds rejected - Table Full  during load
Highest Table Size
Current record count
Storage Alloc (K)  or Total storage allocated
Total storage in-use
Entry storage allocated
Entry storage in-use
Index storage allocated
Index storage in-use
Data storage allocated
Data storage in-use
Rereads
```

Because these are internal fields, the traced values do not always correspond exactly to those in a statistics record.

Chapter 9. Investigating problems

This chapter contains Diagnosis, Modification, or Tuning Information.

It describes the trace and dump information that is produced by CICS to help you determine the cause of a problem with data tables under these headings:

Explanations of the diagnostic messages and abend codes produced by SDT services are contained in the *VSE/ESA Messages and Codes Volume 3* manual.

Using trace information

The trace table produced by CICS helps you to determine the cause of a problem. It shows the flow of control through the CICS modules. This section describes the entries included in the trace table by data table services. For information on the contents of the trace table and how to obtain it, see the *CICS Diagnosis Reference* manual.

There are two types of trace point:

- Entry and exit trace points for each of the services provided by SDT services. File control level-2 tracing must be on to obtain these trace points.
- Exception trace points.

Both of these types are listed separately below.

Entry and exit trace points

The following entry and exit trace points are provided by SDT services:

0B13 Entry to Remote Read service
0B14 Exit from Remote Read service
0B1B Entry to Initialize Data Table Support service
0B1C Exit from Initialize Data Table Support service
0B1D Entry to Logon service
0B1E Exit from Logon service
0B1F Entry to Load service
0B20 Exit from Load service
0B21 Entry to Open, Close, Set Enablement and Statistics services
0B22 Exit from Open, Close, Set Enablement and Statistics services
0B23 Entry to local read services
0B24 Exit from local read services
0B25 Entry to update (add record, add, replace, delete) services
0B26 Exit from update services
0B2D Entry to Connect and Disconnect services
0B2E Exit from Connect and Disconnect services

The format of each of these trace points is described in the *CICS Trace Entries*.

Function and qualifier flags

Each entry and exit trace point contains a function field, and most of them contain a qualifier flags field. The function field is a byte that identifies the function that was being performed; the qualifier flags field is a byte that contains flags that qualify some of the functions. The values of these fields are:

Table 9 (Page 1 of 2). Function and qualifier flags and values

Function	Qualifier flags
X'00' Initialize	X'00' as shared data table server X'80' as shared data table requester
X'02' Add entry from source	X'00' add issued as a result of a data set to table read request X'40' add issued by load transaction
X'03' Write entry to table	X'00' completed write X'80' pre-write for CMT
X'04' Rewrite entry in table	X'00' completed rewrite X'80' pre-rewrite for CMT
X'05' Delete entry in table	X'00' completed delete X'80' pre-delete for CMT
X'06' Commit user-maintained data table updates made by this unit of work	
X'07' Roll back user-maintained data table updates made by this unit of work	
X'08' Load data table (on exit trace only)	X'00' load OK X'80' source file is empty
X'09' Point at a record	X'80' equal match X'40' greater than match X'20' less than match (the above can be in various combinations) X'10' test if data table is enabled
X'0A' Retrieve record by key	X'80' equal match X'40' greater than match X'20' less than match (the above can be in various combinations) X'10' test if data table is enabled
X'0B' Retrieve record by token	X'80' equal match (internal fastpath for a sequence of records) X'40' greater than match X'20' less than match (the above can be in various combinations) X'10' test if data table is enabled
X'0C' Logon as a server	
X'0E' Open a data table	
X'0F' Close a data table	
X'10' Collect statistics	

Table 9 (Page 2 of 2). Function and qualifier flags and values	
Function	**Qualifier flags**
X'11' Set enablement state	X'00' enable data table X'80' disable data table X'40' force disablement (always combined with disable)
X'15' Connect to a shared data table	
X'16' Break connection to a shared data table	
X'17' Process the completion of loading	

Response codes

Each exit trace point contains a two-byte response-code and reason-code field. The first byte is the response code, for which the possible values are:

X'01' Successful
X'02' Exception
X'03' Disaster
X'04' Invalid
X'06' Purged

Reason codes

Each exit trace point contains a two-byte response-code and reason-code field. The second byte is the reason code, for which the possible values are:

X'01' Record not in data table
X'02' Duplicate (record already in data table)
X'03' Data table full (already contains the maximum number of records)
X'04' Record rejected by user exit
X'05' Failed to get storage[2]
X'06' Record not in data table (and table is known to be complete)
X'07' Data table service failed[2]
X'08' Not authorized to connect to file[2]
X'09' Resource is not a data table
X'0A' Remote system has not logged on as a server
X'0B' Load request failed[2]
X'0C' Data table is disabled
X'0D' Add request (from DASD) deliberately not processed
X'0E' Record too long
X'0F' Data table token invalid
X'10' Record not in data table (but might be in source data set)
X'11' Data table not closed as other files are still using it
X'12' Reserved
X'13' Record is in data table but not currently valid
X'14' File cannot be closed as it is disabled
X'15' Protocol error[2]
X'16' CICS is not a VSE subsystem
X'17' Not authorized to connect to this file[2]

---

[2] This reason code might have accompanying error code information. The error code is a four-byte field that is also reported in either an error message or an exception trace point. The possible values are described in the *VSE/ESA Messages and Codes Volume 3* manual and in "Analyzing errors from the SVC" on page 63.

X'18' CICS cannot use cross-memory services
X'19' Interface parameter block format not recognized

UMT and other flags

This flag byte is included in the entry trace point on OPEN. The significant bits at open time are:

B'1.......' CMT
B'01......' Recoverable UMT
B'00......' Nonrecoverable UMT

Exception trace points

The following exception trace points are provided by SDT:

AP 0B0A Unrecognized function on call to DFHDTRE

AP 0B0B Unrecognized function on call to DFHDTRR

AP 0B0C Unrecognized function on call to DFHDTUP

AP 0B0D Unrecognized function on call to DFHDTST

AP 0B0E Unrecognized function on call to DFHDTSS

AP 0B0F Unrecognized function on call to DFHDTRC

AP 0B10 Error on initializing record management

AP 0B11 Error on record manager OPEN

AP 0B12 Error on record manager CLOSE

AP 0B15 Unexpected error on call to retrieval PC

AP 0B19 Error calling data tables SVC

AP 0B1A Error calling data tables SVC

AP 0B27 CLOSE could not find table block

AP 0B28 CLOSE could not find file block

AP 0B29 Error calling data tables SVC

AP 0B2A Error calling data table SVC

AP 0B2B XDTRD exit returned invalid record length (that is, it changed the length for a CMT, or increased the length for a UMT)

AP 0B2C Connect index exceeds maximum supported size

AP 0B2F Disastrous error when acquiring storage to pass parameters to loading transaction

The format of each of these trace points is described in the *CICS Trace Entries*. The following two sections contain guidance on interpreting some of the information that is traced.

Analyzing errors from the SVC

Following an error from a call to the data tables SVC, an exception trace point is always made, which includes an error code field identifying the reason for the error. There are three categories of SVC error:

1. Conditions that are expected to occur, such as the remote file on a connect attempt not being a data table, or the remote system not having logged on as a shared data tables server. CICS takes the appropriate action for such conditions, and no diagnostic information is needed.
2. Errors that could be caused by problems in the environment, which it might be possible to correct. For these errors, a message is issued in which the reason code for the error is reported. The explanation of the reason code is included in the explanation of the message in the *VSE/ESA Messages and Codes Volume 3* manual.
3. Errors that indicate some sort of logic problem, or a misuse of the routines, possibly in an attempt to circumvent integrity or security checks. These errors are treated by CICS file control as disastrous errors, resulting in a system dump (if you have enabled such dumping) and, in most cases, in the transaction being abended with an AFCZ ABEND. For these, the value of the response and reason field is normally X'0215'.

Error codes

This section explains the error codes for the third category of errors that is described above. These error codes are seen only in the exception trace entry. The format of the error code is X'*ffaaaaaa*', in which *ff* identifies the type of failure, and *aaaaaa* is additional information provided for some of the failures. The possible values of *ff* for each trace point are described below.

Values for all trace points

The following error codes can occur for the 0B12, 0B19, 0B1A, 0B29, and 0B2A exception trace points:

X'01' A function was specified that requires the caller to be authorized using the CICS AFCB (authorized function control block), but the caller was not authorized.

X'0A' The caller passed an invalid function code.

X'0B' The caller specified an invalid format of SVC call.

X'0C' An invalid parameter list address was passed to the SVC.

X'0D' A function was specified that requires the value passed in register 1 to be 0, but it was not. The additional information contains the low-order three bytes of the value passed.

X'12' A function was specified that requires the caller to be in Key 0 supervisor state, but the caller was not.

Values for 0B12 trace point

The 0B12 exception trace point is issued if an error is returned by the SVC on adding or deleting an access list entry when a shared data table is being closed. In addition to the errors that can occur at all trace points, the following are possible:

X'02' The CICS region has not yet performed SDT initialization (an anchor block for the region has not been created).

X'0E' The specified data space STOKEN is invalid or the caller is not authorized to use it.

X'0F' The CICS region has not completed initialization as a server.

X'13' An attempt to delete an access list entry failed because the specified entry was not created by the data tables SVC.

All other errors result in a message being issued that contains the error code.

Values for 0B19 trace point

The 0B19 exception trace point is issued if an error is returned by the SVC on initializing as a shared data table server. In addition to the errors that can occur at all trace points, the following are possible:

X'02' An attempt was being made to add an access list entry before the CICS region had performed SDT initialization (an anchor block for the region had not yet been created).

X'0E' The specified data space STOKEN is invalid or the caller is not authorized to use it.

X'0F' An attempt was being made to add an access list entry before the CICS region had completed server initialization.

All other errors result in a message being issued that contains the error code.

Values for 0B1A trace point

The 0B1A exception trace point is issued if an error is returned by the SVC on initializing as a shared data table requester. In addition to the errors that can occur at all trace points, the following are possible:

X'05' The CICS region has already initialized as a shared data table requester, but is now running under a different request block from that under which it originally initialized.

All other errors result in a message being issued that contains the error code.

Values for 0B29 trace point

The 0B29 exception trace point is issued if an error is returned by the SVC on logging on as a shared data table server. In addition to the errors that can occur at all trace points, the following are possible:

X'02' The CICS region that is attempting to register (logon) as a server has not yet been initialized (an anchor block for the region has not been created).

X'04' This CICS region has already registered (logged on) as a shared data tables server.

X'0F' The CICS region has not completed server initialization.

X'14' The AFCS anchor block does not exist.

X'15' The CICS security block does not exist.

All other errors result in a message being issued that contains the error code.

Values for 0B2A trace point

If the function code field contains X'15' then the 0B2A exception trace point indicates an error on CONNECT (that is, on attempting to establish a connection to a remote file). In addition to the errors that can occur at all trace points, the following are possible:

X'02' The requesting region has not performed SDT initialization (an anchor block for the region has not been created).

X'03' The requesting region has not completed initialization as a shared data tables requester.

X'05' The CICS region is running under a different request block (RB) from when it initialized as a data table requester. The additional information part of the error code contains the RB address under which the call was made.

X'72' The LINK to the user-replaceable DFHACEE module to find the home address space's security userid has failed. The additional information part of the error code contains two bytes of the ABEND code from the LINK. The response and reason field accompanying this error is X'020B'.

All other errors result in a message being issued that contains the error code.

If the function code field contains X'16' then the 0B2A exception trace point indicates an error on DISCONNECT (that is, on attempting to break the connection to a remote file). In addition to the errors that can occur at all trace points, the following are possible:

X'02' The requesting region has not performed SDT initialization (an anchor block for the region has not been created).

X'03' The requesting region has not completed initialization as a shared data tables requester.

X'05' The CICS region is running under a different request block (RB) from when it initialized as a data table requester. The additional information part of the error code contains the RB address under which the call was made.

X'07' The caller has supplied an invalid index into the vector of file connections. The additional information part of the error code contains the low-order three bytes of the caller's index.

X'10' The specified connection was broken previously and no longer exists. The additional information part of the error code contains the low-order three bytes of the caller's index into the vector of file connections.

All other errors result in a message being issued that contains the error code.

Analyzing errors from cross-memory services

Following an unexpected error from data tables cross-memory services, an **X'0B15' exception trace entry** is made. It includes the response and reason codes and an error code field identifying the cause of the error. Such errors are all caused either by a corruption of the routines or of the system, or by a possible misuse of the routines.

For a response and reason of X'0215', the format of the error code is X'*ffaaaaaa*', in which *ff* identifies the type of failure and *aaaaaa* is additional information provided for some of the failures. The possible values of *ff* are:

X'01' An attempt to locate the CICS AFCB (authorized function control block) made by either the cross-memory retrieval routine or the connect vector lookup routine has failed.

X'02' The requesting CICS region has not yet performed SDT initialization (an anchor block for the region has not yet been created and set up).

X'03' The requesting region has not completed initialization as a shared data tables requester.

X'05' The retrieval request was issued under a request block different from the one that performed initialization as an SDT requester.

X'06' The connect vector entry for the remote file does not contain the correct linkage index.

X'07' The index of the connect vector entry for the remote file is beyond the end of the connect vector.

X'08' The connect vector entry for the remote file is not marked as being in use.

X'09' The cross-memory retrieval routine has not been called using the correct mechanism.

A response and reason of X'0400' means that the function code passed to the record management code running in the server region was an unrecognized value.

Using dump information

Information relevant to data tables is included in a CICS system dump to help you determine the cause of a problem. For information on the contents of dumps and how to obtain them, see the *CICS Problem Determination Guide*.

The major control blocks that are used by SDT are included in the FILE CONTROL area of a formatted dump of the FOR. These control blocks are:

- Data Table Global Area.

  This is also known as the SDT Header Block and it uses the eye-catcher DFHDTHEADER.

- Data Table Base Area.

  This is also known as the SDT Table Block and it uses the eye-catcher DFHDTTABLE.

- Data Table Path Area.

This is also known as the SDT File Block and it uses the eye-catcher DFHDTFILE.

The data table contents are not included in the CICS system dump because the data space in which the data table resides is not part of the CICS address space. If you want to see the contents of the data table, ask the system operator to use the VSE DUMP command to request a dump of the data space DFHDT001 owned by the appropriate CICS startup job.

The operator command `QUERY DSPACE,partition` displays summary information, including the DSPNAMEs, of data spaces owned by that partition. The VSE DUMP command can be used to dump the contents of the data space to either a printer or tape device.

The DUMP command has the format:

```
DUMP DSPACE,dspname,partition,cuu
```

where

- `dspname` is the data space name (DFHDT001 for shared data tables)
- `partition` is the partition in which the CICS file-owning region is running
- `cuu` is the output device

For more information on the VSE DUMP command, see the *VSE/ESA System Control Statements* manual.

Appendix A. Sample user exit programs

This appendix contains Product-sensitive Programming Interface and Associated Guidance Information.

This appendix describes, by means of samples of coding and data definition sequences, the conventions used in user exit programs that are used with SDT in:

- "Sample XDTRD exit program" on page 70
- "Sample XDTAD exit program" on page 79
- "Sample XDTLC exit program" on page 85

These samples are intended only as general guidance and do not define a programming interface.

The exit programs are supplied with CICS in the VSE/ESA production sublibrary, PRD1.BASE.

Copybook DFHXDTDS defines the data tables user exit parameter list, which is used in each of the following samples. DFHXDTDS is shown in Figure 9 on page 43.

Sample XDTRD exit program

```
         TITLE 'DFH$DTRD - Sample XDTRD Global User Exit Program'
*************************************************************************
*                                                                       *
*   MODULE NAME = DFH$DTRD                                              *
*                                                                       *
* DESCRIPTIVE NAME = CICS/VSE Shared Data Tables Sample XDTRD Exit      *
*                                                                       *
* FUNCTION =                                                            *
*   The program selects records for inclusion in a data table.          *
*                                                                       *
* --------------------------------------------------------------------- *
*   NOTE that this program is intended only to DEMONSTRATE the use      *
*   of the data tables user exit XDTRD, and to show the sort of         *
*   information that can be obtained from the exit parameter list.      *
* IT SHOULD BE TAILORED BEFORE BEING USED IN A PRODUCTION ENVIRONMENT   *
* --------------------------------------------------------------------- *
*                                                                       *
*   This global user exit program is invoked, if enabled, when a        *
*   record which has been fetched from the source data set is about     *
*   to be added to the data table.                                      *
*                                                                       *
*   The program can be used both on CICS systems that are using         *
*   Shared Data Tables support and those that are not.                  *
*   It uses a flag that is passed via the data tables parameter list    *
*   to determine whether the exit has been invoked by shared data       *
*   tables support.                                                     *
*                                                                       *
*   The purpose of the program is to demonstrate the use of the         *
*   option to optimize the data table load by skipping over ranges of   *
*   key values that are to be excluded from the table.  This option     *
```

Figure 13 (Part 1 of 9). Sample XDTRD user exit program

```
*   is allowed only for shared data tables, but the program also      *
*   illustrates that individual records can be rejected when the exit *
*   is not invoked by the data table loading transaction, or when     *
*   shared data tables support is not in use.                         *
*                                                                     *
*   If the program has been invoked by shared data tables support     *
*   it checks whether the source data set name passed to it is the    *
*   one defined by the constant EXITDSN.  If so, and the exit has     *
*   been invoked from the loading transaction, the skip-during-load   *
*   option is used to skip (not attempt to load) any records except   *
*   those whose keys start with the two characters in EXITKEY.        *
*                                                                     *
*   If the program has not been invoked by shared data tables, it     *
*   uses the data table name rather than the source DSname to check   *
*   whether this is the file from which records are to be rejected.   *
*   If the table name matches the constant EXITFILE, only records     *
*   whose keys start with the two characters in EXITKEY are accepted  *
*   for inclusion in the table.                                       *
*                                                                     *
*   A number of the actions taken are for illustrative purposes only, *
*   rather than being the recommended way in which to code an XDTRD   *
*   exit program - for example, the program demonstrates how the      *
*   keylength passed to the exit can be used to avoid having to know  *
*   the keylength of the source data set, whereas in practice this    *
*   might well be known;  and the program chooses to reject any       *
*   records which are not presented to it by the loading transaction, *
*   whereas it would be more realistic to accept all records in the   *
*   desired range of keys.                                            *
*                                                                     *
*   It should also be noted that there are other useful things which  *
*   can be done with the XDTRD exit, such as amending the contents of *
*   the records as they are loaded into a user-maintained data table. *
*                                                                     *
*   The trace flag passed to the exit is set ON if File Control (FC)  *
*   level 1 tracing is enabled.                                       *
*                                                                     *
* NOTES :                                                             *
*    DEPENDENCIES = S/390                                             *
*          DFH$DTRD, or an exit program which is based on this        *
*          sample, must be defined on the CSD as a program            *
*          (with DATALOCATION(ANY)).                                  *
*    RESTRICTIONS =                                                   *
*          This program is designed to run on CICS Transaction        *
*          Server for VSE/ESA Release 1.                              *
*          It needs the DFHXDTDS copybook to be available at          *
*          assembly time.                                             *
*    REGISTER CONVENTIONS = see code                                  *
*    MODULE TYPE = Executable                                         *
*    PROCESSOR = Assembler                                            *
*    ATTRIBUTES = Read only, AMODE 31, RMODE ANY                      *
*                                                                     *
*---------------------------------------------------------------------*
```

Figure 13 (Part 2 of 9). Sample XDTRD user exit program

```
*                                                                     *
* ENTRY POINT = DFH$DTRD                                             *
*                                                                     *
*     PURPOSE =                                                       *
*         Described above                                             *
*                                                                     *
*     LINKAGE =                                                       *
*         Called by the user exit handler                             *
*                                                                     *
*     INPUT =                                                         *
*         Standard user exit parameter list DFHUEPAR,                 *
*          addressed by R1 and containing a pointer to the            *
*          Data Tables parameter list                                 *
*                                                                     *
*     OUTPUT =                                                        *
*         Return code placed in R15                                   *
*         When skipping is requested, a skip-key is returned in an    *
*          area whose address is passed in the parameter list         *
*                                                                     *
*     EXIT-NORMAL =                                                   *
*         Return code in R15 can be                                   *
*         UERCDTAC = accept record (include it in the table)          *
*         UERCDTRJ = reject record (omit it from the table)           *
*         UERCDTOP = optimize load by skipping on to specified key    *
*                      (only possible with shared data tables support) *
*                                                                     *
*     EXIT-ERROR =                                                    *
*         None                                                        *
*                                                                     *
*---------------------------------------------------------------------*
*                                                                     *
* EXTERNAL REFERENCES :                                               *
*     ROUTINES = None                                                 *
*     DATA AREAS = None                                               *
*     CONTROL BLOCKS =                                                *
*         User Exit Parameter list for XDTRD: DFHUEPAR                *
*         Data Tables User Exit Parameter List: DT_UE_PLIST           *
*     GLOBAL VARIABLES = None                                         *
*                                                                     *
* TABLES = None                                                       *
*                                                                     *
* MACROS =                                                            *
*         DFHUEXIT to generate the standard user exit parameter list  *
*                  with the extensions for the XDTRD exit point       *
*         DFHUEXIT to declare the XPI (exit programming interface)    *
*         DFHTRPTX XPI call to issue a user trace entry               *
*                                                                     *
*---------------------------------------------------------------------*
```

Figure 13 (Part 3 of 9). Sample XDTRD user exit program

```
*                                                                     *
* DESCRIPTION of the program structure:                               *
*                                                                     *
*    1) Standard entry code for a global user exit that uses the XPI: *
*         The program sets up any definitions required, saves the     *
*         caller's registers, establishes addressability, and         *
*         addresses the parameter lists.                              *
*    2) Initial section of code:                                      *
*         The program gets the key address and length from the data   *
*         table parameter list, tests whether the exit is invoked     *
*         from shared data table code and, if not, branches to the    *
*         non-SDT section of code.                                    *
*   3a) SDT code - Skipping:                                          *
*         If the source data set is the one from which records are to *
*         be selected, and if the exit has been called by the data    *
*         table load, the program compares the record key with a      *
*         value that defines the range included in the data table,    *
*         and sets return codes which will cause the data table load  *
*         to skip over any other keys.                                *
*   3b) SDT code - Tracing:                                           *
*         If FC level 1 tracing is enabled, the program issues a user *
*         trace point X'0128', branches to set R15 and returns.       *
*   4a) non-SDT code - choosing whether to accept record:             *
*         If the file name is the one for which records are to be     *
*         selected, the program rejects the record if it does not     *
*         match the value which defines the range to be included in   *
*         the data table.                                             *
*   4b) non-SDT code - Tracing:                                       *
*         If FC level 1 tracing is enabled, the program issues a user *
*         trace point X'0118'.                                        *
*    5) Set R15 and return:                                           *
*         The program sets a return code in R15 and performs          *
*         standard exit code for a global user exit (restores         *
*         caller's registers, and returns to the address that was in  *
*         R14 when the exit program was called).                      *
*                                                                     *
***********************************************************************
         EJECT ,
         DFHUEXIT TYPE=EP,ID=XDTRD  Standard UE parameters for XDTRD
         DFHUEXIT TYPE=XPIENV       Exit programming interface (XPI)
         EJECT ,
         COPY  DFHXDTDS             Additional data table UE params
         EJECT ,
         COPY  DFHTRPTY             Trace definitions
         EJECT ,
```

Figure 13 (Part 4 of 9). Sample XDTRD user exit program

```
*********************************************************************
*  REGISTER USAGE :                                                 *
*  R0 -                                                             *
*  R1 - address of DFHUEPAR on input, and used by XPI calls         *
*  R2 - address of standard user exit parameter list, DFHUEPAR      *
*  R3 - record length                                               *
*  R4 - address of data set name (SDT) or data table name (non-SDT) *
*  R5 - address of storage for XPI parameters                       *
*  R6 - address of data tables parameter list, DT_UE_PLIST          *
*  R7 - final return code to be set in R15                          *
*  R8 - address of the record key                                   *
*  R9 - key length                                                  *
*  R10- address of the skip-key area (SDT only)                     *
*  R11- base register                                               *
*  R12- address of data table flags byte, UEPDTFLG                  *
*  R13- address of kernel stack prior to XPI CALLS                  *
*  R14- used by XPI calls                                           *
*  R15- return code and used by XPI calls                           *
*  (The register equates are declared by the DFHUEXIT call above)   *
*********************************************************************
         SPACE 2
DFH$DTRD CSECT
DFH$DTRD AMODE 31
DFH$DTRD RMODE ANY
         STM   R14,R12,12(R13)     Save callers registers
         LR    R11,R15             Set up base register
         USING DFH$DTRD,R11
         LR    R2,R1               Address standard parameters
         USING DFHUEPAR,R2
         L     R6,UEPDTPL          Address data table parameters
         USING DT_UE_PLIST,R6
         L     R8,UEPDTKA          Key address
         L     R9,UEPDTKL          Key length
*********************************************************************
* Test whether the exit was invoked from shared data tables support  *
*********************************************************************
         TM    UEPDTFLG,UEPDTSDT   Were we invoked from SDT?
         BZ    NOTSDT              Branch to non-SDT code if not
         EJECT ,
*********************************************************************
*        Invoked from SDT, so can use skip optimization             *
*********************************************************************
         L     R10,UEPDTSKA        Get skip-key area address
*********************************************************************
* Determine whether the source data set is the one on which          *
* optimization by skipping is to be performed.                       *
* If it is not, just accept all records.                             *
*********************************************************************
         CLC   UEPDTDSN,EXITDSN
         BNE   SDTACC
```

Figure 13 (Part 5 of 9). Sample XDTRD user exit program

```
*********************************************************************
* Only keys in the range defined by the initial two characters in   *
* EXITKEY are to be accepted, any other ranges of key values        *
* are to be skipped.                                                *
* First check whether skipping is valid - skipping can only be used *
* when the call has been issued by the loading transaction (which is*
* indicated by the UEPDTOPT flag being set).                        *
*********************************************************************
         TM    UEPDTFLG,UEPDTOPT   Can we skip?
         BZ    SDTREJ              Just reject rec if not (although  *
                                    in a production version it would *
                                    make more sense to check whether *
                                    the record key is in the desired *
                                    range, and accept it if so)
*********************************************************************
* EXITKEY is currently set to 'AD', so that the effect of           *
* the exit will be:                                                 *
* - If the key is less than 'AD....' then skip to a skip-key of     *
*   'AD' padded with 00s.                                           *
* - If it starts with 'AD' then accept it.                          *
* - If it is greater than 'AD' then skip to a skip-key of 'FF's     *
* If the value of the constant EXITKEY is altered, the program causes*
* the new range it defines to be selected for inclusion in the table.*
* Note that if a different length of EXITKEY is needed to define the*
* range to be accepted, the code also needs amendment, as it        *
* currently assumes a length of 2.                                  *
*********************************************************************
         CLC   0(2,R8),EXITKEY     Is key below or above 'AD' ?
         BE    SDTACC              If equal then just accept
         BH    HIGHER              Above so skip to end of file
         SPACE 1
LOWER    DS    0H                  Skip forwards to 'AD...'
         MVC   0(2,R10),EXITKEY    Set skip-key value
         LR    R15,R9              Keylength for padding
         SH    R15,=H'3'            minus 2 for 'AD' and 1 for XC
         EX    R15,XCSKP           Clear out rest of skip key
         LA    R7,UERCDTOP         Indicate skipping
         B     SDTTR
         SPACE 1
HIGHER   DS    0H                  Skip on to end of file
         MVI   0(R10),X'FF'        Set 'FF' in start of skip key
         LR    R15,R9              Get length of skip-key
         BCTR  R15,0               Decrement for pad length
         BCTR  R15,0               Decrement for MVC
         EX    R15,MVCSKP          Propagate 'FF' through skip-key
         LA    R7,UERCDTOP         Indicate skipping
         B     SDTTR
         SPACE 1
```

Figure 13 (Part 6 of 9). Sample XDTRD user exit program

```
*********************************************************************
* Store return code in R7 for accept or reject                      *
*********************************************************************
SDTACC   DS    0H
         LA    R7,UERCDTAC         Indicate accept
         B     SDTTR
SDTREJ   DS    0H
         LA    R7,UERCDTRJ         Indicate reject
         EJECT ,
*********************************************************************
*        Tracing when SDT support is in use                         *
*********************************************************************
SDTTR    L     R15,UEPTRACE
         TM    0(R15),UEPTRON      Is trace on?
         BZ    NOSDTTR             No - do not issue trace then
         L     R5,UEPXSTOR         Prepare for XPI call
         USING DFHTRPT_ARG,R5
         L     R13,UEPSTACK
*********************************************************************
* Trace source data set name, key, record length, flags, and skip-key *
* Some of these fields are available only with SDT support           *
*********************************************************************
         LA    R4,UEPDTDSN         Point at data set name
         LA    R3,UEPDTRL          Address of record length for trace
         LA    R12,UEPDTFLG        Point at the data table flags
         DFHTRPTX CALL,                                             *
               CLEAR,                                               *
               IN,                                                  *
               FUNCTION(TRACE_PUT),                                 *
               POINT_ID(RDTRACE2),                                  *
               DATA1((R4),UEPDTDSL),                                *
               DATA2((R8),(R9)),                                    *
               DATA3((R3),4),                                       *
               DATA4((R12),1),                                      *
               DATA5((R10),(R9)),                                   *
               OUT,                                                 *
               RESPONSE(*),                                         *
               REASON(*)
NOSDTTR  DS    0H
         B     FINISH              Go and return from exit
         EJECT ,
```

Figure 13 (Part 7 of 9). Sample XDTRD user exit program

```
*********************************************************************
*        Exit was NOT invoked by shared data tables support         *
*********************************************************************
NOTSDT   DS    0H
*********************************************************************
* Determine whether this data table is the one from which records    *
* are to be rejected.  If it is not, just accept all records.        *
*********************************************************************
         CLC   UEPDTNAM,EXITFILE
         BNE   ACC
         CLC   0(2,R8),EXITKEY     Does key start with 'AD' ?
         BE    ACC                 Yes, so accept it
REJ      DS    0H
         LA    R7,UERCDTRJ         Indicate reject
         B     TRACE               Go and issue trace
ACC      DS    0H
         LA    R7,UERCDTAC         Indicate accept
         SPACE 2
*********************************************************************
*        Tracing when SDT support is not in use                     *
*********************************************************************
TRACE    L     R15,UEPTRACE
         TM    0(R15),UEPTRON      Is trace on?
         BZ    NOTRACE             No - do not issue trace then
         L     R5,UEPXSTOR         Prepare for XPI call
         USING DFHTRPT_ARG,R5
         L     R13,UEPSTACK
*********************************************************************
* Trace data table name, key, record length, and flags               *
*********************************************************************
         LA    R4,UEPDTNAM         Point at data table name
         LA    R3,UEPDTRL          Address of reclen for trace
         LA    R12,UEPDTFLG        Point at the data table flags
         DFHTRPTX CALL,                                             *
               CLEAR,                                               *
               IN,                                                  *
               FUNCTION(TRACE_PUT),                                 *
               POINT_ID(RDTRACE1),                                  *
               DATA1((R4),8),                                       *
               DATA2((R8),(R9)),                                    *
               DATA3((R3),4),                                       *
               DATA4((R12),1),                                      *
               OUT,                                                 *
               RESPONSE(*),                                         *
               REASON(*)
```

Figure 13 (Part 8 of 9). Sample XDTRD user exit program

```
NOTRACE  DS    0H
         EJECT ,
*------------------------------------------------------------------*
*        Code to set R15 return code and return control            *
*------------------------------------------------------------------*
FINISH   DS    0H
         LR    R15,R7              Pick up exit return code
         L     R13,UEPEPSA         Standard GLUE ending code
         L     R14,12(R13)
         LM    R0,R12,20(R13)
         BR    R14
         SPACE 2
*------------------------------------------------------------------*
*        Constants and executed instructions                       *
*------------------------------------------------------------------*
EXITDSN  DC    CL44'CFV23.CSYSW1.SOURCED'
EXITFILE DC    CL8'CICD'
EXITKEY  DC    C'AD'
         SPACE 1
RDTRACE1 DC    XL2'118'
RDTRACE2 DC    XL2'128'
         SPACE 1
MVCSKP   MVC   1(*-*,R10),0(R10)   Executed to propagate X'FF's
XCSKP    XC    2(*-*,R10),2(R10)   Executed to clear to X'00's
         SPACE 1
         END   DFH$DTRD
```

Figure 13 (Part 9 of 9). Sample XDTRD user exit program

Sample XDTAD exit program

```
         TITLE 'DFH$DTAD - Sample XDTAD Global User Exit Program'
*************************************************************************
*                                                                       *
*   MODULE NAME = DFH$DTAD                                              *
*                                                                       *
* DESCRIPTIVE NAME = CICS/VSE Shared Data Tables Sample XDTAD Exit      *
*                                                                       *
* FUNCTION =                                                            *
*   The program selects records for inclusion in a shared data table. *
*                                                                       *
* --------------------------------------------------------------------- *
*   NOTE that this program is intended only to DEMONSTRATE the use      *
*   of the data tables user exit XDTAD, and to show the sort of         *
*   information that can be obtained from the exit parameter list.      *
* IT SHOULD BE TAILORED BEFORE BEING USED IN A PRODUCTION ENVIRONMENT *
* --------------------------------------------------------------------- *
*                                                                       *
*   This global user exit program is invoked, if enabled, when a        *
*   WRITE request is issued to a data table.                            *
*                                                                       *
*   The program can be used both on CICS systems that are using         *
*   Shared Data Tables support, and those that are not.                 *
*   It uses a flag that is passed via the data tables parameter list    *
*   to determine whether the exit has been invoked by shared data       *
*   tables support.                                                     *
*                                                                       *
*   The purpose of the program is to demonstrate the use of the XDTAD   *
*   global user exit to select only certain records for inclusion in    *
*   a data table.  In this example, the selection is made on the        *
*   basis of key values.  If shared data tables support is being        *
*   used, and the source data set for the data table is that            *
*   specified by the constant EXITDSN, the program selects particular   *
*   keys for inclusion in the data table, and reject others.  For all   *
*   other source data sets, or if shared data table support is not in   *
*   use, all records are accepted.                                      *
*                                                                       *
*   The record selection is made on the basis of the value of the 6th   *
*   character in the record key.  This is for illustrative purposes     *
*   only, as it is unlikely to be the criterion for selection in a      *
*   realistic environment.  For example, for a shared CICS-maintained   *
*   data table, it might be desirable to select a group of records      *
*   which are known to be very frequently read by applications          *
*   running in other CICS regions in the MVS system.                    *
*                                                                       *
*   The trace flag passed to the exit is set ON if File Control (FC)    *
*   level 1 tracing is enabled.                                         *
*                                                                       *
```

Figure 14 (Part 1 of 6). Sample XDTAD user exit program

```
* NOTES :                                                              *
*    DEPENDENCIES = S/390                                              *
*          DFH$DTAD, or an exit program that is based on this          *
*          sample, must be defined on the CSD as a program             *
*          (with DATALOCATION(ANY)).                                   *
*    RESTRICTIONS =                                                    *
*          This program is designed to run on CICS Transaction         *
*          Server for VSE/ESA Release 1.                               *
*          It needs the DFHXDTDS copybook to be available at           *
*          assembly time.                                              *
*    REGISTER CONVENTIONS = see code                                   *
*    MODULE TYPE = Executable                                          *
*    PROCESSOR = Assembler                                             *
*    ATTRIBUTES = Read only, AMODE 31, RMODE ANY                       *
*                                                                      *
*----------------------------------------------------------------------*
*                                                                      *
* ENTRY POINT = DFH$DTAD                                               *
*                                                                      *
*     PURPOSE =                                                        *
*         Described above                                              *
*                                                                      *
*     LINKAGE =                                                        *
*         Called by the user exit handler                              *
*                                                                      *
*     INPUT =                                                          *
*         Standard user exit parameter list DFHUEPAR,                  *
*          addressed by R1 and containing a pointer to the             *
*          Data Tables parameter list                                  *
*                                                                      *
*     OUTPUT =                                                         *
*         Return code placed in R15                                    *
*                                                                      *
*     EXIT-NORMAL =                                                    *
*         Return code in R15 can be                                    *
*         UERCDTAC = accept record (include it in the table)           *
*         UERCDTRJ = reject record (omit it from the table)            *
*                                                                      *
*     EXIT-ERROR =                                                     *
*         None                                                         *
*                                                                      *
*----------------------------------------------------------------------*
*                                                                      *
* EXTERNAL REFERENCES :                                                *
*     ROUTINES = None                                                  *
*     DATA AREAS = None                                                *
*     CONTROL BLOCKS =                                                 *
*         User Exit Parameter list for XDTAD: DFHUEPAR                 *
*         Data Tables User Exit Parameter List: DT_UE_PLIST            *
*     GLOBAL VARIABLES = None                                          *
*                                                                      *
* TABLES = None                                                        *
*                                                                      *
```

Figure 14 (Part 2 of 6). Sample XDTAD user exit program

```
* MACROS =                                                             *
*         DFHUEXIT to generate the standard user exit parameter list  *
*                  with the extensions for the XDTAD exit point       *
*         DFHUEXIT to declare the XPI (exit programming interface)    *
*         DFHTRPTX XPI call to issue a user trace entry               *
*                                                                     *
*---------------------------------------------------------------------*
*                                                                     *
* DESCRIPTION of the program structure:                               *
*                                                                     *
*    1) Standard entry code for a global user exit that uses the XPI: *
*         The program sets up any definitions required, saves the     *
*         caller's registers, establishes addressability, and         *
*         addresses the parameter lists.                              *
*    2) Initial section of code:                                      *
*         The program gets the key address and length from the data   *
*         table parameter list, and tests whether the exit was        *
*         invoked from shared data table code.  If not, it branches   *
*         to the non-SDT section of code.                             *
*   3a) SDT code - Tracing:                                           *
*         If FC level 1 tracing is enabled, the program issues a user *
*         trace point X'0129', and branches to choose whether to      *
*         accept the record for inclusion in the table.               *
*   3b) non-SDT code - Tracing:                                       *
*         If FC level 1 tracing is enabled, the program issues a user *
*         trace point X'0119', accepts the record, and exits.         *
*    4) SDT code - choosing whether to accept record:                 *
*         If the source data set is the one that records are selected *
*         from, the program, if the key contains a numeric character  *
*         in the sixth byte, sets the return code so that the record  *
*         is accepted.  If the key contains an alphabetic character   *
*         in the sixth byte, the program sets a return code to reject *
*         the record.                                                 *
*    5) Standard exit code for a global user exit that uses the XPI:  *
*         The program restores users registers, and returns to the    *
*         address that was in R14 when the exit program was called.   *
*                                                                     *
***********************************************************************
         EJECT ,
         DFHUEXIT TYPE=EP,ID=XDTAD  Standard UE parameters for XDTAD
         DFHUEXIT TYPE=XPIENV       Exit programming interface (XPI)
         EJECT ,
         COPY  DFHXDTDS             Additional data table UE params
         EJECT ,
         COPY  DFHTRPTY             Trace definitions
         EJECT ,
```

Figure 14 (Part 3 of 6). Sample XDTAD user exit program

```
*********************************************************************
*  REGISTER USAGE :                                                 *
*  R0 -                                                             *
*  R1 - address of DFHUEPAR on input, and used by XPI calls         *
*  R2 - address of standard user exit parameter list, DFHUEPAR      *
*  R3 -                                                             *
*  R4 - address of source data set name                             *
*  R5 - address of storage for XPI parameters                       *
*  R6 - address of data tables parameter list, DT_UE_PLIST          *
*  R7 - address of the trace flag                                   *
*  R8 - address of the record key                                   *
*  R9 - key length                                                  *
*  R10- address of the data table name                              *
*  R11- base register                                               *
*  R12- address of data table flags byte, UEPDTFLG                  *
*  R13- address of kernel stack prior to XPI CALLS                  *
*  R14- used by XPI calls                                           *
*  R15- return code, and used by XPI calls                          *
*  (The register equates are declared by the DFHUEXIT call above)   *
*********************************************************************
         SPACE 2
DFH$DTAD CSECT
DFH$DTAD AMODE 31
DFH$DTAD RMODE ANY
         STM   R14,R12,12(R13)     Save callers registers
         LR    R11,R15             Set up base register
         USING DFH$DTAD,R11
         LR    R2,R1               Address standard parameters
         USING DFHUEPAR,R2
         L     R6,UEPDTPL          Address data table parameters
         USING DT_UE_PLIST,R6
*********************************************************************
* Save some fields from the parameter list that are to be traced or *
* used in selecting records                                         *
*********************************************************************
         L     R8,UEPDTKA          Key address
         L     R9,UEPDTKL          Key length
         LA    R10,UEPDTNAM        Data table name
*********************************************************************
* Test whether the exit was invoked from shared data tables support *
*********************************************************************
         TM    UEPDTFLG,UEPDTSDT   Were we invoked from SDT?
         BZ    NOTSDT              Branch to non-SDT code if not
         EJECT ,
*********************************************************************
*        Invoked from SDT, so can use SDT fields in parameter list. *
*        Issue trace, then check data set name, and select records. *
*********************************************************************
         L     R7,UEPTRACE         Address of trace flag
         TM    0(R7),UEPTRON       Is trace on?
         BZ    NOSDTTR             No - do not issue trace then
         L     R5,UEPXSTOR         Prepare for XPI call
         USING DFHTRPT_ARG,R5
         L     R13,UEPSTACK
```

Figure 14 (Part 4 of 6). Sample XDTAD user exit program

```
*********************************************************************
* Trace key, data table name, source data set name, and flags.      *
* The last two fields are only meaningful for SDT support.          *
*********************************************************************
         LA    R4,UEPDTDSN         Point at the source data set name
         LA    R12,UEPDTFLG        Point at the data table flags
         DFHTRPTX CALL,                                            *
               CLEAR,                                              *
               IN,                                                 *
               FUNCTION(TRACE_PUT),                                *
               POINT_ID(ADTRACE2),                                 *
               DATA1((R8),(R9)),                                   *
               DATA2((R10),8),                                     *
               DATA3((R4),UEPDTDSL),                               *
               DATA4((R12),1),                                     *
               OUT,                                                *
               RESPONSE(*),                                        *
               REASON(*)
NOSDTTR  B     CHOOSE              Go and choose whether to accept rec
         EJECT ,
*********************************************************************
*        Exit has not been invoked from SDT                         *
*        Issue trace then accept the record.                        *
*********************************************************************
NOTSDT   L     R7,UEPTRACE         Address of trace flag
         TM    0(R7),UEPTRON       Is trace on?
         BZ    NOTRACE             No - do not issue trace then
         L     R5,UEPXSTOR         Prepare for XPI call
         USING DFHTRPT_ARG,R5
         L     R13,UEPSTACK
*********************************************************************
* Trace key and data table name                                     *
*********************************************************************
         DFHTRPTX CALL,                                            *
               CLEAR,                                              *
               IN,                                                 *
               FUNCTION(TRACE_PUT),                                *
               POINT_ID(ADTRACE1),                                 *
               DATA1((R8),(R9)),                                   *
               DATA2((R10),8),                                     *
               OUT,                                                *
               RESPONSE(*),                                        *
               REASON(*)
         SPACE 1
```

Figure 14 (Part 5 of 6). Sample XDTAD user exit program

```
NOTRACE  DS    0H
         B     ACC                Go and accept the record
         EJECT ,
*************************************************************************
*        Is this the data set from which records are to be selected?  *
*        If not, just accept record and end.                          *
*************************************************************************
CHOOSE   DS    0H
         CLC   UEPDTDSN,EXITDSN
         BNE   ACC
*************************************************************************
*                                                                       *
* If the sixth character in the key is numeric, accept the record;    *
* if it is alphabetic, reject the record.                              *
* This assumes that numerics have EBCDIC value >= F0                   *
*          and that alphabetics have     value <  F0                   *
*                                                                       *
*************************************************************************
         CLI   5(R8),X'F0'        Is sixth character >= F0 ?
         BL    REJ                If no, then go and reject record
ACC      LA    R15,UERCDTAC       If yes, set RC to ACCEPT
         B     GLUEND              and end
REJ      LA    R15,UERCDTRJ       Set RC to REJECT
         SPACE 3
GLUEND   DS    0H                 Standard GLUE ending code
         L     R13,UEPEPSA
         L     R14,12(R13)
         LM    R0,R12,20(R13)
         BR    R14
         SPACE 2
*************************************************************************
*        Constants                                                      *
*************************************************************************
EXITDSN  DC    CL44'CFV23.CSYSW1.SOURCED'
         SPACE 1
ADTRACE1 DC    XL2'119'
ADTRACE2 DC    XL2'129'
         SPACE 2
         END   DFH$DTAD
```

Figure 14 (Part 6 of 6). Sample XDTAD user exit program

Sample XDTLC exit program

```
        TITLE 'DFH$DTLC - Sample XDTLC Global User Exit Program'
*****************************************************************
*                                                               *
*   MODULE NAME = DFH$DTLC                                      *
*                                                               *
* DESCRIPTIVE NAME = CICS/VSE Shared Data Tables Sample XDTLC Exit *
*                                                               *
* FUNCTION =                                                    *
*   The program rejects a data table if its load did not complete OK. *
*                                                               *
* ------------------------------------------------------------- *
*   NOTE that this program is intended only to DEMONSTRATE the use *
*   of the data tables user exit XDTLC, and to show the sort of  *
*   information that can be obtained from the exit parameter list. *
* IT SHOULD BE TAILORED BEFORE BEING USED IN A PRODUCTION ENVIRONMENT *
* ------------------------------------------------------------- *
*                                                               *
*   This global user exit program is invoked, if enabled, when the *
*   load of a data table has completed.                         *
*                                                               *
*   The program can be used both on CICS systems that are using  *
*   Shared Data Tables support, and those that are not.         *
*   It uses a flag that is passed via the data tables parameter list *
*   to determine whether the exit has been invoked by shared data *
*   tables support.                                             *
*                                                               *
*   The program issues a user trace entry if tracing is enabled, and *
*   checks the setting of the load completion indicator.        *
*   If this shows that loading failed to complete successfully, the *
*   exit program sets a return code that rejects the table by   *
*   requesting that it is closed.                               *
*                                                               *
*   The trace flag passed to the exit is set ON if File Control (FC) *
*   level 1 tracing is enabled.                                 *
*                                                               *
* NOTES :                                                       *
*    DEPENDENCIES = S/390                                       *
*          DFH$DTLC, or an exit program that is based on this   *
*          sample, must be defined on the CSD as a program      *
*          (with DATALOCATION(ANY)).                            *
*    RESTRICTIONS =                                             *
*          This program is designed to run on CICS Transaction  *
*          Server for VSE/ESA Release 1.
*          It needs the DFHXDTDS copybook to be available at    *
*          assembly time.                                       *
*    REGISTER CONVENTIONS = see code                            *
*    MODULE TYPE = Executable                                   *
*    PROCESSOR = Assembler                                      *
*    ATTRIBUTES = Read only, AMODE 31, RMODE ANY                *
*                                                               *
*---------------------------------------------------------------*
```

Figure 15 (Part 1 of 5). Sample XDTLC user exit program

```
*                                                                     *
* ENTRY POINT = DFH$DTLC                                             *
*                                                                     *
*     PURPOSE =                                                       *
*         Described above                                             *
*                                                                     *
*     LINKAGE =                                                       *
*         Called by the user exit handler                             *
*                                                                     *
*     INPUT =                                                         *
*         Standard user exit parameter list DFHUEPAR,                 *
*          addressed by R1 and containing a pointer to the            *
*          Data Tables parameter list                                 *
*                                                                     *
*     OUTPUT =                                                        *
*         Return code placed in R15                                   *
*                                                                     *
*     EXIT-NORMAL =                                                   *
*         Return code in R15 can be                                   *
*         UERCDTOK = accept table                                     *
*         UERCDTCL = reject table (close it)                          *
*                                                                     *
*     EXIT-ERROR =                                                    *
*         None                                                        *
*                                                                     *
*---------------------------------------------------------------------*
*                                                                     *
* EXTERNAL REFERENCES :                                              *
*     ROUTINES = None                                                 *
*     DATA AREAS = None                                               *
*     CONTROL BLOCKS =                                                *
*         User Exit Parameter list for XDTLC: DFHUEPAR                *
*         Data Tables User Exit Parameter List: DT_UE_PLIST           *
*     GLOBAL VARIABLES = None                                         *
*                                                                     *
* TABLES = None                                                      *
*                                                                     *
* MACROS =                                                           *
*         DFHUEXIT to generate the standard user exit parameter list  *
*                  with the extensions for the XDTLC exit point       *
*         DFHUEXIT to declare the XPI (exit programming interface)    *
*         DFHTRPTX XPI call to issue a user trace entry               *
*                                                                     *
*---------------------------------------------------------------------*
*                                                                     *
* DESCRIPTION of the program structure:                              *
*                                                                     *
*    1) Standard entry code for a global user exit that uses the XPI: *
*         The program sets up any definitions required, saves the     *
*         caller's registers, establishes addressability, and         *
*         addresses the parameter lists.                              *
```

Figure 15 (Part 2 of 5). Sample XDTLC user exit program

```
*    2) Tracing (only executed if FC level 1 tracing is enabled):     *
*         The program tests whether it was invoked from shared data   *
*         table code.  If so, it issues a user trace point X'0126'    *
*         including fields from the data table parameter list that    *
*         are supplied only by shared data table support.  If not, it *
*         issues a X'0116' trace point, containing parameters which   *
*         are supplied by any level of data table support.            *
*    3) Choosing whether to accept the table:                         *
*         The program tests the return code from the load.  If the    *
*         load failed to complete, it sets UERCDTCL in R15, which     *
*         requests that the table is closed.  If the load completed   *
*         successfully, it sets UERCDTOK in R15 to keep the table     *
*         open.                                                       *
*    4) Standard exit code for a global user exit that uses the XPI:  *
*         The program restores users registers, and returns to the    *
*         address that was in R14 when the exit program was called.   *
*                                                                     *
***********************************************************************
         EJECT ,
         DFHUEXIT TYPE=EP,ID=XDTLC  standard UE parameters for XDTLC
         DFHUEXIT TYPE=XPIENV       exit programming interface (XPI)
         EJECT ,
         COPY  DFHXDTDS             Additional data table UE params
         EJECT ,
         COPY  DFHTRPTY             Trace definitions
         EJECT ,
***********************************************************************
*  Register usage :                                                   *
*  R0 -                                                               *
*  R1 - address of DFHUEPAR on input, and used by XPI calls           *
*  R2 - address of standard user exit plist, DFHUEPAR                 *
*  R3 -                                                               *
*  R4 - address of source data set name                               *
*  R5 - address of storage for XPI parameters                         *
*  R6 - address of data tables parameter list, DT_UE_PLIST            *
*  R7 - address of the trace flag, UEPTRACE                           *
*  R8 - address of data table name                                    *
*  R9 - address of loading completion indicator, UEPDTORC             *
*  R10-                                                               *
*  R11- base register                                                 *
*  R12- address of data table flags byte, UEPDTFLG                    *
*  R13- address of kernel stack prior to XPI calls                    *
*  R14- used by XPI calls                                             *
*  R15- return code, and used by XPI calls                            *
*  (The register equates are declared by the DFHUEXIT call above)     *
***********************************************************************
```

Figure 15 (Part 3 of 5). Sample XDTLC user exit program

```
         SPACE 2
DFH$DTLC CSECT
DFH$DTLC AMODE 31
DFH$DTLC RMODE ANY
         STM   R14,R12,12(R13)     Save caller's registers
         LR    R11,R15             Establish base
         USING DFH$DTLC,R11
         LR    R2,R1               Address standard parameters
         USING DFHUEPAR,R2
         L     R6,UEPDTPL          Address data table parameters
         USING DT_UE_PLIST,R6
         SPACE 1
         LA    R8,UEPDTNAM         Address data table name
         LA    R9,UEPDTORC         Address load return code
*********************************************************************
*        Issue Trace (if tracing is enabled)                        *
*********************************************************************
         L     R7,UEPTRACE         Get trace flag address
         TM    0(R7),UEPTRON       Is trace on?
         BZ    CHOOSE              Skip tracing if not
*********************************************************************
* Test whether the exit was invoked from shared data tables support   *
*********************************************************************
         TM    UEPDTFLG,UEPDTSDT   Were we invoked from SDT?
         BZ    NOTSDT              Branch if not
         EJECT ,
*********************************************************************
*        Exit has been invoked from SDT                             *
*********************************************************************
         L     R5,UEPXSTOR         Set up XPI trace call
         USING DFHTRPT_ARG,R5
         L     R13,UEPSTACK
*********************************************************************
* Trace data table name, load return code, source dsname, and flags.  *
* The last two fields are only meaningful for SDT support.            *
*********************************************************************
         LA    R4,UEPDTDSN         Get source data set name
         LA    R12,UEPDTFLG        Get data table flags
         DFHTRPTX CALL,                                             *
               CLEAR,                                               *
               IN,                                                  *
               FUNCTION(TRACE_PUT),                                 *
               POINT_ID(LCTRACE2),                                  *
               DATA1((R8),8),                                       *
               DATA2((R9),1),                                       *
               DATA3((R4),UEPDTDSL),                                *
               DATA4((R12),1),                                      *
               OUT,                                                 *
               RESPONSE(*),                                         *
               REASON(*)
         B     CHOOSE              Go and test if load completed OK
         EJECT ,
```

Figure 15 (Part 4 of 5). Sample XDTLC user exit program

```
*********************************************************************
*        Exit has not been invoked from SDT                         *
*********************************************************************
NOTSDT   L     R5,UEPXSTOR         Set up XPI trace call
         USING DFHTRPT_ARG,R5
         L     R13,UEPSTACK
*********************************************************************
* Trace data table name and load return code                        *
*********************************************************************
         DFHTRPTX CALL,                                            *
               CLEAR,                                              *
               IN,                                                 *
               FUNCTION(TRACE_PUT),                                *
               POINT_ID(LCTRACE1),                                 *
               DATA1((R8),8),                                      *
               DATA2((R9),1),                                      *
               OUT,                                                *
               RESPONSE(*),                                        *
               REASON(*)
         EJECT ,
*********************************************************************
*        If load completed successfully, keep table open.           *
*        If not, ask for it to be closed                            *
*********************************************************************
CHOOSE   DS    0H
         CLI   0(R9),UEPDTLFL      Did load fail?
         BNE   LOADOK
         LA    R15,UERCDTCL        Set RC for table to be closed
         B     GLUEND
LOADOK   LA    R15,UERCDTOK        Set RC to keep table
         SPACE 3
GLUEND   DS    0H                  Standard GLUE exit code
         L     R13,UEPEPSA
         L     R14,12(R13)
         LM    R0,R12,20(R13)
         BR    R14
         SPACE 2
*********************************************************************
* Constant Declarations                                             *
*********************************************************************
LCTRACE1 DC    XL2'116'
LCTRACE2 DC    XL2'126'
         SPACE 1
         END   DFH$DTLC
```

Figure 15 (Part 5 of 5). Sample XDTLC user exit program

Bibliography

CICS Transaction Server for VSE/ESA Release 1 library

Evaluation and planning	
Release Guide	GC33-1645
Migration Guide	GC33-1646
Report Controller Planning Guide	SC33-1941
General	
Master Index	SC33-1648
Trace Entries	SX33-6108
User's Handbook	SX33-6101
Glossary (softcopy only)	GC33-1649
Administration	
System Definition Guide	SC33-1651
Customization Guide	SC33-1652
Resource Definition Guide	SC33-1653
Operations and Utilities Guide	SC33-1654
CICS-Supplied Transactions	SC33-1655
Programming	
Application Programming Guide	SC33-1657
Application Programming Reference	SC33-1658
Sample Applications Guide	SC33-1713
Application Migration Aid Guide	SC33-1943
System Programming Reference	SC33-1659
Distributed Transaction Programming Guide	SC33-1661
Front End Programming Interface User's Guide	SC33-1662
Diagnosis	
Problem Determination Guide	GC33-1663
Messages and Codes Vol 3 (softcopy only)	SC33-6799
Diagnosis Reference	LY33-6085
Data Areas	LY33-6086
Supplementary Data Areas	LY33-6087
Communication	
Intercommunication Guide	SC33-1665
CICS Family: Interproduct Communication	SC33-0824
CICS Family: Communicating from CICS on System/390	SC33-1697
Special topics	
Recovery and Restart Guide	SC33-1666
Performance Guide	SC33-1667
Shared Data Tables Guide	SC33-1668
Security Guide	SC33-1942
External CICS Interface	SC33-1669
XRF Guide	SC33-1671
Report Controller User's Guide	SC34-5688
CICS Clients	
CICS Clients: Administration	SC33-1792
CICS Universal Clients Version 3 for OS/2: Administration	SC34-5450
CICS Universal Clients Version 3 for Windows: Administration	SC34-5449
CICS Universal Clients Version 3 for AIX: Administration	SC34-5348
CICS Universal Clients Version 3 for Solaris: Administration	SC34-5451
CICS Family: OO programming in C++ for CICS Clients	SC33-1923
CICS Family: OO programming in BASIC for CICS Clients	SC33-1671
CICS Family: Client/Server Programming	SC33-1435
CICS Transaction Gateway Version 3: Administration	SC34-5448

Books from VSE/ESA 2.4 base program libraries

VSE/ESA Version 2 Release 4

Book title	Order number
Administration	SC33-6705
Diagnosis Tools	SC33-6614
Extended Addressability	SC33-6621
Guide for Solving Problems	SC33-6710
Guide to System Functions	SC33-6711
Installation	SC33-6704
Licensed Program Specification	GC33-6700
Messages and Codes Volume 1	SC33-6796
Messages and Codes Volume 2	SC33-6798
Messages and Codes Volume 3	SC33-6799
Networking Support	SC33-6708
Operation	SC33-6706
Planning	SC33-6703
Programming and Workstation Guide	SC33-6709
System Control Statements	SC33-6713
System Macro Reference	SC33-6716
System Macro User's Guide	SC33-6715
System Upgrade and Service	SC33-6702
System Utilities	SC33-6717
TCP/IP User's Guide	SC33-6601
Turbo Dispatcher Guide and Reference	SC33-6797
Unattended Node Support	SC33-6712

High-Level Assembler Language (HLASM)

Book title	Order number
General Information	GC26-8261
Installation and Customization Guide	SC26-8263
Language Reference	SC26-8265
Programmer's Guide	SC26-8264

Language Environment for VSE/ESA (LE/VSE)

Book title	Order number
C Run-Time Library Reference	SC33-6689
C Run-Time Programming Guide	SC33-6688
Concepts Guide	GC33-6680
Debug Tool for VSE/ESA Fact Sheet	GC26-8925
Debug Tool for VSE/ESA Installation and Customization Guide	SC26-8798
Debug Tool for VSE/ESA User's Guide and Reference	SC26-8797
Debugging Guide and Run-Time Messages	SC33-6681
Diagnosis Guide	SC26-8060
Fact Sheet	GC33-6679
Installation and Customization Guide	SC33-6682
LE/VSE Enhancements	SC33-6778
Licensed Program Specification	GC33-6683
Programming Guide	SC33-6684
Programming Reference	SC33-6685
Run-Time Migration Guide	SC33-6687
Writing Interlanguage Communication Applications	SC33-6686

VSE/ICCF

Book title	Order number
Adminstration and Operations	SC33-6738
User's Guide	SC33-6739

VSE/POWER

Book title	Order number
Administration and Operation	SC33-6733
Application Programming	SC33-6736
Networking Guide	SC33-6735
Remote Job Entry User's Guide	SC33-6734

VSE/VSAM

Book title	Order number
Commands	SC33-6731
User's Guide and Application Programming	SC33-6732

VTAM for VSE/ESA

Book title	Order number
Customization	LY43-0063
Diagnosis	LY43-0065
Data Areas	LY43-0104
Messages and Codes	SC31-6493
Migration Guide	GC31-8072
Network Implementation Guide	SC31-6494
Operation	SC31-6495
Overview	GC31-8114
Programming	SC31-6496
Programming for LU6.2	SC31-6497
Release Guide	GC31-8090
Resource Definition Reference	SC31-6498

Books from VSE/ESA 2.4 optional program libraries

C for VSE/ESA (C/VSE)

Book title	Order number
C Run-Time Library Reference	SC33-6689
C Run-Time Programming Guide	SC33-6688
Diagnosis Guide	GC09-2426
Installation and Customization Guide	GC09-2422
Language Reference	SC09-2425
Licensed Program Specification	GC09-2421
Migration Guide	SC09-2423
User's Guide	SC09-2424

COBOL for VSE/ESA (COBOL/VSE)

Book title	Order number
Debug Tool for VSE/ESA Fact Sheet	GC26-8925
Debug Tool for VSE/ESA Installation and Customization Guide	SC26-8798
Debug Tool for VSE/ESA User's Guide and Reference	SC26-8797
Diagnosis Guide	SC26-8528
General Information	GC26-8068
Installation and Customization Guide	SC26-8071
Language Reference	SC26-8073
Licensed Program Specifications	GC26-8069
Migration Guide	GC26-8070
Migrating VSE Applications To Advanced COBOL	GC26-8349
Programming Guide	SC26-8072

DB2 Server for VSE

Book title	Order number
Application Programming	SC09-2393
Database Administration	GC09-2389
Installation	GC09-2391
Interactive SQL Guide and Reference	SC09-2410
Operation	SC09-2401
Overview	GC08-2386
System Administration	GC09-2406

DL/I VSE

Book title	Order number
Application and Database Design	SH24-5022
Application Programming: CALL and RQDLI Interface	SH12-5411
Application Programming: High-Level Programming Interface	SH24-5009
Database Administration	SH24-5011
Diagnostic Guide	SH24-5002
General Information	GH20-1246
Guide for New Users	SH24-5001
Interactive Resource Definition and Utilities	SH24-5029
Library Guide and Master Index	GH24-5008
Licensed Program Specifications	GH24-5031
Low-level Code and Continuity Check Feature	SH20-9046
Library Guide and Master Index	GH24-5008
Messages and Codes	SH12-5414
Recovery and Restart Guide	SH24-5030
Reference Summary: CALL Program Interface	SX24-5103
Reference Summary: System Programming	SX24-5104
Reference Summary: HLPI Interface	SX24-5120
Release Guide	SC33-6211

PL/I for VSE/ESA (PL/I VSE)

Book title	Order number
Compile Time Messages and Codes	SC26-8059
Debug Tool For VSE/ESA User's Guide and Reference	SC26-8797
Diagnosis Guide	SC26-8058
Installation and Customization Guide	SC26-8057
Language Reference	SC26-8054
Licensed Program Specifications	GC26-8055
Migration Guide	SC26-8056
Programming Guide	SC26-8053
Reference Summary	SX26-3836

Screen Definition Facility II (SDF II)

Book title	Order number
VSE Administrator's Guide	SH12-6311
VSE General Introduction	SH12-6315
VSE Primer for CICS/BMS Programs	SH12-6313
VSE Run-Time Services	SH12-6312

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply in the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

This publication could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM United Kingdom Laboratories, MP151, Hursley Park, Winchester, Hampshire, England, SO21 2JN. Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Programming License Agreement, or any equivalent agreement between us.

Trademarks and service marks

The following terms, used in this publication, are trademarks or service marks of IBM Corporation in the United States or other countries:

CICS
CICS/VSE
IBM
VSE/ESA

Programming interface information

This book is intended to help you set up and use CICS shared data tables. This book also documents General-use Programming Interface and Associated Guidance Information, Product-sensitive Programming Interface and Associated Guidance Information, and Diagnosis, Modification, or Tuning Information that is provided by CICS.

General-use programming interfaces allow the customer to write programs that obtain the services of CICS.

Programming Interface and Associated Guidance Information is identified where it occurs by an introductory statement to a chapter or section.

Product-sensitive programming interfaces allow the customer installation to perform tasks such as diagnosing, modifying, monitoring, repairing, tailoring, or tuning of CICS. Use of such interfaces creates dependencies on the detailed design or implementation of the IBM® software product. Product-sensitive programming interfaces should be used only for these specialized purposes. Because of their dependencies on detailed design and implementation, it is to be expected that programs written to such interfaces may need to be changed in order to run with new product releases or versions, or as a result of service.

Product-Sensitive Programming Interface and Associated Guidance Information is identified where it occurs by an introductory statement to a chapter or section.

Diagnosis, Modification, or Tuning Information is provided to help you diagnose problems and tailor your CICS system.

Warning: Do not use this Diagnosis, Modification, or Tuning Information as a programming interface.

Diagnosis, Modification, or Tuning Information is identified where it occurs by an introductory statement to a chapter or section.

Index

E

F

G

I

J

K

L

M

N

U

V

X

Sending your comments to IBM

CICS® Transaction Server for VSE/ESA™

Shared Data Tables Guide

SC33-1668-00

If you want to send to IBM any comments you have about this book, please use one of the methods listed below. Feel free to comment on anything you regard as a specific error or omission in the subject matter, and on the clarity, organization or completeness of the book itself.

To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate, without incurring any obligation to you.

You can send your comments to IBM in any of the following ways:

- By mail:

  IBM UK Laboratories
  Information Development
  Mail Point 095
  Hursley Park
  Winchester, SO21 2JN
  England

- By fax:
  - From outside the U.K., after your international access code use 44 1962 870229
  - From within the U.K., use 01962 870229
- Electronically, use the appropriate network ID:
  - IBM Mail Exchange: GBIBM2Q9 at IBMMAIL
  - IBMLink: HURSLEY(IDRCF)
  - Email: idrcf@hursley.ibm.com

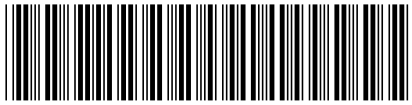
Whichever method you use, ensure that you include:

- The publication number and title
- The page number or topic to which your comment applies
- Your name and address/telephone number/fax number/network ID.

Program Number: 5648-054

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC33-1668-00



Spine information:

CICS TS for VSE/ESA Shared Data Tables Guide *Release 1*